PASTA PUBLIC

SCRIPS.JS

const currentPage = location.pathname

const menuItems = document.querySelectorAll("header .links a")

for(item of menuItems){

    if(currentPage.includes(item.getAttribute("href"))){

        item.classList.add('active')

    }

}

//Paginação

// totalPages = 20

// selectedPage=6

// [1,...,4,5,6,7,8,...,20]

// selectedPage=15

// [1,...,13,14,15,16,17,...,20]

function paginate (*selectedPage*, *totalPages*){

    let pages=[],

        oldPage

    for(let currentPage = 1; currentPage<=totalPages; currentPage++) {

        const firstAndLastPage = currentPage ==1 || currentPage == totalPages

        const pagesAfterSelectedPage = currentPage<= selectedPage +2

        const pagesBeforeSelectedPage =currentPage>= selectedPage -2

        if(firstAndLastPage || pagesAfterSelectedPage && pagesBeforeSelectedPage){

            if(oldPage && currentPage - oldPage >2){

                pages.push("...")

            }

            if(oldPage && currentPage - oldPage == 2){

                pages.push(oldPage+1)

            }

            pages.push(currentPage)

            oldPage = currentPage

        }

    }

    return pages

}

function createPagination(*pagination*){

    const filter = pagination.dataset.filter

    const page = +pagination.dataset.page

    const total = +pagination.dataset.total

    const pages = paginate(page, total)

    let elements =""

    for (let page of pages){

        if(*String*(page).includes("...")){

            elements += `<span>${page}</span>`

        } else{

            if(filter){

                elements += `<a href="?page=${page}&filter=${filter}">${page}</a>`

            }else{

                elements += `<a href="?page=${page}">${page}</a>`

            }

        }

    }

    pagination.innerHTML = elements

}

const pagination = document.querySelector(".pagination")

if(pagination){

    createPagination(pagination)

}

STYLE.CSS

PASTA SRC

PASTA SRC/APP

PASTA SRC/APP/CONTROLLERS

INSTRUCTORS.JS

const {age, date} = require('../../lib/utils')

const Instructor = require('../models/Instructor')

*module*.*exports* = {

    index(*req*, *res*){

        let {filter, page, limit} = req.query

        page = page || 1

        limit = limit || 2

        let offset = limit\*(page-1)

        const params ={

            filter,

            page,

            limit,

            offset,

            callback(*instructors*){

                const pagination = {

                    total: Math.ceil(instructors[0].total/limit),

                    page

                }

                return res.render("instructors/index", {instructors, pagination, filter})

            }

        }

        Instructor.paginate(params)

    },

    create(*req*, *res*){

        return res.render("instructors/create")

    },

    post(*req*, *res*){

        const keys = *Object*.keys(req.body)

        for(key of keys){

            if(req.body[key] == "") {

                return res.send('Please, fill all fields!')

            }

        }

        Instructor.create(req.body, function(*instructor*){

            return res.redirect(`/instructors/${instructor.id}`)

        })

    },

    show(*req*, *res*){

        Instructor.find(req.params.id, function(*instructor*){

            if(!instructor) return res.send("Instructor not found!")

            instructor.age = age(instructor.birth)

            instructor.services = instructor.services.split(",")

            instructor.created\_at = date(instructor.created\_at).format

            return res.render("instructors/show", {instructor})

        })

    },

    edit(*req*, *res*){

        Instructor.find(req.params.id, function(*instructor*){

            if(!instructor) return res.send("Instructor not found!")

            instructor.birth = date(instructor.birth).iso

            instructor.services = instructor.services.split(",")

            instructor.created\_at = date(instructor.created\_at).format

            return res.render("instructors/edit", {instructor})

        })

    },

    put(*req*, *res*){

        const keys = *Object*.keys(req.body)

        for(key of keys){

            if(req.body[key] == "") {

                return res.send('Please, fill all fields!')

            }

        }

        Instructor.update(req.body, function(){

            return res.redirect(`/instructors/${req.body.id}`)

        })

    },

    delete(*req*, *res*){

        Instructor.delete(req.body.id, function(){

            return res.redirect(`/instructors`)

        })

    },

}

MEMBERS.JS

const {age, date} = require('../../lib/utils')

const Member = require('../models/Member')

*module*.*exports* = {

    index(*req*, *res*){

        let {filter, page, limit} = req.query

        page = page || 1

        limit = limit || 2

        let offset = limit\*(page-1)

        const params ={

            filter,

            page,

            limit,

            offset,

            callback(*members*){

                const pagination = {

                    total: Math.ceil(members[0].total/limit),

                    page

                }

                return res.render("members/index", {members, pagination, filter})

            }

        }

        Member.paginate(params)

    },

    create(*req*, *res*){

        Member.instructorsSelectOptions(function(*options*){

            return res.render('members/create', { instructorOptions: options})

        })

    },

    post(*req*, *res*){

        const keys = *Object*.keys(req.body)

        for(key of keys){

            if(req.body[key] == "") {

                return res.send('Please, fill all fields!')

            }

        }

        Member.create(req.body, function(*member*){

            return res.redirect(`/members/${member.id}`)

        })

    },

    show(*req*, *res*){

        Member.find(req.params.id, function(*member*){

            if(!member) return res.send("Member not found!")

            member.birth = date(member.birth).birtDay

            return res.render("members/show", {member})

        })

    },

    edit(*req*, *res*){

        Member.find(req.params.id, function(*member*){

            if(!member) return res.send("Member not found!")

            member.birth = date(member.birth).iso

            Member.instructorsSelectOptions(function(*options*){

                return res.render('members/edit', { member, instructorOptions: options})

            })

        })

    },

    put(*req*, *res*){

        const keys = *Object*.keys(req.body)

        for(key of keys){

            if(req.body[key] == "") {

                return res.send('Please, fill all fields!')

            }

        }

        Member.update(req.body, function(){

            return res.redirect(`/members/${req.body.id}`)

        })

    },

    delete(*req*, *res*){

        Member.delete(req.body.id, function(){

            return res.redirect(`/members`)

        })

    },

}

PASTA SRC/APP/MODELS

Instructor.js

const {date} = require('../../lib/utils')

const db = require('../../config/db')

*module*.*exports* = {

    all(*callback*){

        db.query(`

        SELECT instructors.\*, count(members) AS total\_students

        FROM instructors

        LEFT JOIN members ON (members.instructor\_id = instructors.id)

        GROUP BY instructors.id

        ORDER BY total\_students DESC`, function (*err*, *results*){

            if(err) throw `Database Error! ${err}`

            callback(results.rows)

        })

    },

    create(*data*, *callback*){

        const query = `

            INSERT INTO instructors(

                name,

                avatar\_url,

                gender,

                services,

                birth,

                created\_at

            ) VALUES ($1, $2, $3, $4, $5, $6)

            RETURNING id

        `

        const values = [

            data.name,

            data.avatar\_url,

            data.gender,

            data.services,

            date(data.birth).iso,

            date(*Date*.now()).iso,

        ]

        db.query(query, values, function(*err*, *results*){

            if(err) throw `Database Error! ${err}`

            callback(results.rows[0])

        })

    },

    find(*id*, *callback*){

        db.query(`SELECT \* FROM instructors WHERE id = $1`, [id], function (*err*, *results*){

            if(err) throw `Database Error! ${err}`

            callback(results.rows[0])

        })

    },

    findBy(*filter*, *callback*){

        db.query(`

        SELECT instructors.\*, count(members) AS total\_students

        FROM instructors

        LEFT JOIN members ON (members.instructor\_id = instructors.id)

        WHERE instructors.name ILIKE '%${filter}%'

        OR instructors.services ILIKE '%${filter}%'

        GROUP BY instructors.id

        ORDER BY total\_students DESC`, function (*err*, *results*){

            if(err) throw `Database Error! ${err}`

            callback(results.rows)

        })

    },

    update(*data*, *callback*) {

        const query = `

        UPDATE instructors SET

            avatar\_url =($1),

            name =($2),

            birth =($3),

            gender =($4),

            services =($5)

        WHERE id =$6

        `

        const values =[

            data.avatar\_url,

            data.name,

            date(data.birth).iso,

            data.gender,

            data.services,

            data.id

        ]

        db.query(query, values, function(*err*, *results*){

            if(err) throw `Database Error! ${err}`

            callback()

        })

    },

    delete(*id*, *callback*){

        db.query(`DELETE FROM instructors WHERE id=$1`, [id], function(*err*, *results*){

            if(err) throw `Database Error! ${err}`

            return callback()

        })

    },

    paginate(*params*){

        const {filter, limit, offset, callback } = params

        let query="",

            filterQuery = "",

            totalQuery = `(SELECT count(\*) FROM instructors) AS total`

        if(filter){

            filterQuery =`

            WHERE instructors.name ILIKE '%${filter}%'

            OR instructors.services ILIKE '%${filter}%'

            `

            totalQuery =`( SELECT count(\*) FROM instructors ${filterQuery}) AS total`

        }

query = `SELECT instructors.\*, ${totalQuery}, count(members) AS tot al\_students FROM instructors

            LEFT JOIN members ON (instructors.id = members.instructor\_id)

            ${filterQuery}

            GROUP BY instructors.id LIMIT $1 OFFSET $2`

        db.query(query, [limit, offset], function(*err*, *results*){

            if (err) throw 'Database Error!'

            callback(results.rows)

        })

    }

}

Member.js

const {date} = require('../../lib/utils')

const db = require('../../config/db')

*module*.*exports* = {

    all(*callback*){

        db.query(`SELECT \* FROM members`, function (*err*, *results*){

            if(err) throw `Database Error! ${err}`

            callback(results.rows)

        })

    },

    create(*data*, *callback*){

        const query = `

            INSERT INTO members(

                name,

                avatar\_url,

                gender,

                email,

                birth,

                blood,

                weight,

                height,

                instructor\_id

            ) VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9)

            RETURNING id

        `

        const values = [

            data.name,

            data.avatar\_url,

            data.gender,

            data.email,

            date(data.birth).iso,

            data.blood,

            data.weight,

            data.height,

            data.instructor

        ]

        db.query(query, values, function(*err*, *results*){

            if(err) throw `Database Error! ${err}`

            callback(results.rows[0])

        })

    },

    find(*id*, *callback*){

        db.query(`

            SELECT members.\*, instructors.name AS instructor\_name

            FROM members

            LEFT JOIN instructors ON (members.instructor\_id = instructors.id)

            WHERE members.id = $1`, [id], function (*err*, *results*){

            if(err) throw `Database Error! ${err}`

            callback(results.rows[0])

        })

    },

    update(*data*, *callback*) {

        const query = `

        UPDATE members SET

            avatar\_url =($1),

            name =($2),

            birth =($3),

            gender =($4),

            email =($5),

            blood =($6),

            weight =($7),

            height =($8),

            instructor\_id = ($9)

        WHERE id =$10

        `

        const values =[

            data.avatar\_url,

            data.name,

            date(data.birth).iso,

            data.gender,

            data.email,

            data.blood,

            data.weight,

            data.height,

            data.instructor,

            data.id

        ]

        db.query(query, values, function(*err*, *results*){

            if(err) throw `Database Error! ${err}`

            callback()

        })

    },

    delete(*id*, *callback*){

        db.query(`DELETE FROM members WHERE id=$1`, [id], function(*err*, *results*){

            if(err) throw `Database Error! ${err}`

            return callback()

        })

    },

    instructorsSelectOptions(*callback*){

        db.query(`SELECT name, id FROM instructors`, function(*err*, *results*){

            if(err) throw `Database Error! ${err}`

            callback(results.rows)

        })

    },

    paginate(*params*){

        const {filter, limit, offset, callback } = params

        let query="",

            filterQuery = "",

            totalQuery = `(SELECT count(\*) FROM members) AS total`

        if(filter){

            filterQuery =`

            WHERE members.name ILIKE '%${filter}%'

            OR members.email ILIKE '%${filter}%'

            `

            totalQuery =`( SELECT count(\*) FROM members ${filterQuery}) AS total`

        }

        query = `

            SELECT members.\*, ${totalQuery}

            FROM members

            ${filterQuery}

            LIMIT $1 OFFSET $2`

        db.query(query, [limit, offset], function(*err*, *results*){

            if (err) throw 'Database Error!'

            callback(results.rows)

        })

    }

}

PASTA SRC/APP/VIEWS

PASTA SRC/APP/VIEWS/INSTRUCTORS

create.njk

{% extends "layout.njk" %}

{% block content %}

    <form *class*="card" *method*="POST" *action*="/instructors" >

        <section *class*="avatar" *style*="background: url(https://source.unsplash.com/collection/3465564/500x500) no-repeat center center /cover">

        </section>

        <section *class*="details">

            <h3>Novo Instrutor</h3>

        {% include 'instructors/fields.njk'%}

        </section>

    </form>

{% endblock content %}

edit.njk

{% extends "layout.njk" %}

{% block content %}

    <div *class*="card">

        <section

*class*="avatar"

*style*="background: url({{instructor.avatar\_url}}) no-repeat center center /cover">

        </section>

        <section *class*="details">

            <form *method*="POST" *action*="/instructors?\_method=PUT">

                <h3>Editar Instrutor</h3>

                {% include 'instructors/fields.njk'%}

                <input

*type*="hidden"

*name*="id"

*value*= {{instructor.id}}>

            </form>

            <form *id*="form-delete" *action*="/instructors?\_method=DELETE" *method*="post" >

                <input

*type*="hidden"

*name*="id"

*value*={{instructor.id}}>

                <button *type*="submit">Deletar</button>

            </form>

        </section>

    </div>

    <script>

        const formDelete = document.querySelector("#form-delete")

        formDelete.addEventListener("submit", function(*event*){

            const confirmation = confirm("Deseja deletar?")

            if(!confirmation){

            event.preventDefault()

            }

        })

    </script>

{% endblock content %}

fields.njk

<div *class*="item">

    <div>Avatar URL</div>

    <div>

        <input *type*="url" *name*="avatar\_url" *placeholder*="http://" *value*="{{instructor.avatar\_url}}" >

    </div>

    </div>

    <div *class*="item">

        <div>Instrutor</div>

        <div>

            <input *type*="text" *name*="name" *placeholder*="Nome do Instrutor" *value*="{{instructor.name}}"  >

        </div>

    </div>

    <div *class*="item">

        <div>Data de Nascimento</div>

        <div><input *type*="date" *name*="birth" *value*="{{instructor.birth}}"></div>

    </div>

    <div *class*="item">

        <div>sexo</div>

        <div>

            <span>

                <input

*type*="radio"

*name*="gender"

*value*="M"

*{%* *if* *instructor.gender* =*=* "M" *%}*

*checked*

*{%* *endif* *%}*

                    >Masculino

            </span>

            <span>

                <input

*type*="radio"

*name*="gender"

*value*="F"

*{%* *if* *instructor.gender* =*=* "F" *%}*

*checked*

*{%* *endif* *%}*

                    >Feminino

            </span>

        </div>

    </div>

    <div *class*="item">

        <div>Área de atuação</div>

        <div>

            <input *type*="text" *name*="services" *placeholder*="Separe os serviços prestados por vírgula" *value*="{{instructor.services}}">

        </div>

    </div>

    <button *type*="submit">Salvar</button>

index.njk

{% extends "layout.njk" %}

{% block content %}

    <div *class*="card table-container">

        <div *class*="header">

            <a *class*="button" *href*="/instructors/create">Novo</a>

            <form *action*="/instructors" *method*="GET">

                <input *type*="text" *name*="filter"*placeholder*="Nome ou Atendimento" *value*="{{filter}}">

                <button *type*="submit">Filtrar</button>

            </form>

        </div>

        <table *width*="100%">

            <thead>

                <tr>

                    <th>Instrutor</th>

                    <th>Atendimento</th>

                    <th>Alunos</th>

                    <th>Ação</th>

                </tr>

            </thead>

            <tbody>

                {% for instructor in instructors %}

                <tr>

                    <td>

                        <span *style*="background-image: url({{instructor.avatar\_url}})" ></span>

                        {{instructor.name}}

                    </td>

                    <td>{{instructor.services}}</td>

                    <td>{{instructor.total\_students}}</td>

                    <td><a *href*="/instructors/{{instructor.id}}">ver</a></td>

                </tr>

                {% endfor %}

            </tbody>

        </table>

        <div *class*="pagination"

*data-page* = {{pagination.page}}

*data-total*={{pagination.total}}

*data-filter*={{filter}}

        ></div>

    </div>

{% endblock content %}

show.njk

{% extends "layout.njk" %}

{% block content %}

    <div *class*="card">

        <section *class*="avatar" *style*="background: url({{instructor.avatar\_url}}) no-repeat center center /cover">

        </section>

        <section *class*="details">

            <h3>Detalhes</h3>

        <div *class*="item">

            <div>Instrutor</div>

            <div>{{instructor.name}}</div>

        </div>

        <div *class*="item">

            <div>Idade</div>

            <div>{{instructor.age}}</div>

        </div>

        <div *class*="item">

            <div>sexo</div>

            <div>{{"Masculino" if instructor.gender =="M" else "Feminino"}}</div>

        </div>

        <div *class*="item">

            <div>Acompanhamento</div>

            <div>

                {% for service in instructor.services %}

                    <span>{{ service }}</span>

                {%endfor%}

            </div>

        </div>

        <div *class*="item">

            <div>Desde</div>

            <div>{{instructor.created\_at}}</div>

        </div>

        <a *href*="/instructors/{{instructor.id}}/edit" *class*="button">Editar</a>

        </section>

    </div>

{% endblock content %}

PASTA SRC/APP/VIEWS/MEMBERS

create.njk

{% extends "layout.njk" %}

{% block content %}

    <form *class*="card" *method*="POST" *action*="/members" >

        <section *class*="avatar" *style*="background: url(https://source.unsplash.com/collection/3465564/500x500) no-repeat center center /cover">

        </section>

        <section *class*="details">

            <h3>Novo Membro</h3>

        {% include 'members/fields.njk'%}

        </section>

    </form>

{% endblock content %}

edit.njk

{% extends "layout.njk" %}

{% block content %}

    <div *class*="card">

        <section

*class*="avatar"

*style*="background: url({{member.avatar\_url}}) no-repeat center center /cover">

        </section>

        <section *class*="details">

            <form *method*="POST" *action*="/members?\_method=PUT">

                <h3>Editar Membro</h3>

                {% include 'members/fields.njk'%}

                <input

*type*="hidden"

*name*="id"

*value*={{member.id}}>

            </form>

            <form *id*="form-delete" *action*="/members?\_method=DELETE" *method*="POST">

                <input

*type*="hidden"

*name*="id"

*value*={{member.id}}>

                <button *type*="submit">Deletar</button>

            </form>

        </section>

    </div>

    <script>

        const formDelete = document.querySelector("#form-delete")

        formDelete.addEventListener("submit", function(*event*){

            const confirmation = confirm("Deseja deletar?")

            if(!confirmation){

            event.preventDefault()

            }

        })

    </script>

{% endblock content %}

fields.njk

<div *class*="item">

    <div>Avatar URL</div>

    <div>

        <input *type*="url" *name*="avatar\_url" *placeholder*="http://" *value*="{{member.avatar\_url}}" >

    </div>

    </div>

    <div *class*="item">

        <div>Nome</div>

        <div>

            <input *type*="text" *name*="name" *value*="{{member.name}}"  >

        </div>

    </div>

    <div *class*="item">

        <div>Email</div>

        <div>

            <input *type*="email" *name*="email" *value*="{{member.email}}"  >

        </div>

    </div>

    <div *class*="item">

        <div>Data de Nascimento</div>

        <div><input *type*="date" *name*="birth" *value*="{{member.birth}}"></div>

    </div>

    <div *class*="item">

        <div>sexo</div>

        <div>

            <span>

                <input

*type*="radio"

*name*="gender"

*value*="M"

*{%* *if* *member.gender* =*=* "M" *%}*

*checked*

*{%* *endif* *%}*

                    >Masculino

            </span>

            <span>

                <input

*type*="radio"

*name*="gender"

*value*="F"

*{%* *if* *member.gender* =*=* "F" *%}*

*checked*

*{%* *endif* *%}*

                    >Feminino

            </span>

        </div>

    </div>

    <div *class*="item">

        <div>Tipo Sanguíneo</div>

        <div>

            <select *name*="blood">

                <option *value*="A+"

*{%* *if* *member.blood* =*=*"A+"*%}selected{%endif%}*>A+

                </option>

                <option *value*="A-"

*{%* *if* *member.blood* =*=*"A-"*%}selected{%endif%}*>A-

                </option>

                <option *value*="B+"

*{%* *if* *member.blood* =*=*"B+"*%}selected{%endif%}*>B+

                </option>

                <option *value*="B-"

*{%* *if* *member.blood* =*=*"B-"*%}selected{%endif%}*>B-

                </option>

                <option *value*="AB+"

*{%* *if* *member.blood* =*=*"AB+"*%}selected{%endif%}*>AB

                +</option>

                <option *value*="AB-"

*{%* *if* *member.blood* =*=*"AB-"*%}selected{%endif%}*>AB

                -</option>

                <option *value*="O+"

*{%* *if* *member.blood* =*=*"O+"*%}selected{%endif%}*>O+

                </option>

                <option *value*="O-"

*{%* *if* *member.blood* =*=*"O-"*%}selected{%endif%}*>O-

                </option>

            </select>

        </div>

    </div>

    <div *class*="item">

        <div>Peso(kg)</div>

        <div>

            <input

*type*="number"

*name*="weight"

*value*="{{member.weight}}">

        </div>

    </div>

    <div *class*="item">

        <div>Altura(cm)</div>

        <div>

            <input

*type*="number"

*name*="height"

*value*="{{member.height}}">

        </div>

    </div>

    <div *class*="item">

        <div>Instrutor</div>

        <div>

            <select *name*="instructor">

                {% for instructor in instructorOptions %}

                    <option *value*="{{instructor.id}}"

*{%* *if* *member.instructor\_id* =*=* instructor.id *%}selected{%endif%}*>{{instructor.name}}

                    </option>

                {%endfor%}

            </select>

        </div>

    </div>

    <button *type*="submit">Salvar</button>

index.njk

{% extends "layout.njk" %}

{% block content %}

    <div *class*="card table-container">

         <div *class*="header">

            <a *class*="button" *href*="/members/create">Novo</a>

            <form *action*="/members" *method*="GET">

                <input *type*="text" *name*="filter"*placeholder*="Nome ou email" *value*="{{filter}}">

                <button *type*="submit">Filtrar</button>

            </form>

        </div>

        <table *width*="100%">

            <thead>

                <tr>

                    <th>Membror</th>

                    <th>Email</th>

                    <th>Altura</th>

                    <th>Peso</th>

                    <th>Ação</th>

                </tr>

            </thead>

            <tbody>

                {% for member in members %}

                <tr>

                    <td>

                        <span *style*="background-image: url({{member.avatar\_url}})" ></span>

                        {{member.name}}</td>

                    <td>{{member.email}}</td>

                    <td>{{member.weight}}kg</td>

                    <td>{{member.height}}cm</td>

                    <td><a *href*="/members/{{member.id}}">ver</a></td>

                </tr>

                {% endfor %}

            </tbody>

        </table>

        <div *class*="pagination"

*data-page* = {{pagination.page}}

*data-total*={{pagination.total}}

*data-filter*={{filter}}

        ></div>

    </div>

{% endblock content %}

show.njk

{% extends "layout.njk" %}

{% block content %}

    <div *class*="card">

        <section *class*="avatar" *style*="background: url({{member.avatar\_url}}) no-repeat center center /cover">

        </section>

        <section *class*="details">

            <h3>Detalhes</h3>

        <div *class*="item">

            <div>Membro</div>

            <div>{{member.name}}</div>

        </div>

        <div *class*="item">

            <div>Email</div>

            <div>{{member.email}}</div>

        </div>

        <div *class*="item">

            <div>Aniversário</div>

            <div>{{member.birth}}</div>

        </div>

        <div *class*="item">

            <div>sexo</div>

          <div>{{"Masculino" if member.gender =="M" else "Feminino"}}</div>

        </div>

        <div *class*="item">

            <div>Tipo Sanguíneo</div>

            <div>{{member.blood}}</div>

        </div>

        <div *class*="item">

            <div>Peso(kg)</div>

            <div>{{member.weight}}</div>

        </div>

        <div *class*="item">

            <div>Altura(cm)</div>

            <div>{{member.height}}</div>

        </div>

        <div *class*="item">

            <div>Aluno de </div>

            <div>{{member.instructor\_name}}</div>

        </div>

        <a *href*="/members/{{member.id}}/edit" *class*="button">Editar</a>

        </section>

    </div>

{% endblock content %}

PASTA SRC/VIEWS

Layout.njk

<!DOCTYPE *html*>

<html *lang*="pt-br">

    <head>

        <meta *charset*="UTF-8">

        {% block head%}

            <title>Gym Manager</title>

        {% endblock%}

        <link *rel*="stylesheet" *href*="/styles.css">

    </head>

    <body>

        <header>

            <div *class*="links">

                <a *href*="/instructors">Instructors</a>

                <a *href*="/members">Members</a>

            </div>

        </header>

        {% block content %}{% endblock %}

        <script *src*="/scripts.js"></script>

    </body>

</html>

PASTA SRC/CONFIG

db.js

const {Pool} = require("pg")

*module*.*exports* = **new** *Pool*({

    user:'postgres',

    password: "capota",

    host: "localhost",

    port: 5432,

    database:"gymmanager"

})

PASTA SRC/LIB

utils.js

*module*.*exports* ={

    age(*timestamp*){

        const today = **new** *Date*()

        const birthDate = **new** *Date*(timestamp)

         let age = today.getFullYear() - birthDate.getFullYear()

        const month = today.getMonth() - birthDate.getMonth()

        if(month<0||month==0 && today.getDate()<= birthDate.getDate()){

            age = age -1        }

        return age

    },

    date(*timestamp*){

        const date = **new** *Date*(timestamp)

        const year = date.getUTCFullYear()

        const month = `0${date.getUTCMonth()+1}`.slice(-2)

        const day = `0${date.getUTCDate()}`.slice(-2)

        return {

            day,

            month,

            year,

            iso:`${year}-${month}-${day}`,

            birtDay:`${day}/${month}`,

            format: `${day}/${month}/${year}`

        }

    }

}

PASTA SRC

routes.js

const express = require('express')

const routes = express.Router()

const instructors = require('./app/controllers/instructors')

const members = require('./app/controllers/members')

routes.get('/', function (*req*, *res*){

    return res.redirect("/instructors")

})

routes.get('/instructors', instructors.index)

routes.get('/instructors/create', instructors.create)

routes.get('/instructors/:id', instructors.show)

routes.get('/instructors/:id/edit', instructors.edit)

routes.post("/instructors", instructors.post)

routes.put("/instructors", instructors.put)

routes.delete("/instructors", instructors.delete)

routes.get('/members', members.index)

routes.get('/members/create', members.create)

routes.get('/members/:id', members.show)

routes.get('/members/:id/edit', members.edit)

routes.post("/members", members.post)

routes.put("/members", members.put)

routes.delete("/members", members.delete)

*module*.*exports* = routes

server.js

const express = require('express')

const nunjucks = require('nunjucks')

const routes = require('./routes')

const methodOverride = require('method-override')

const server = express()

server.use(express.urlencoded({extended:true}))

server.use(express.static('public'))

server.use(methodOverride('\_method'))

server.use(routes)

server.set("view engine", "njk")

nunjucks.configure("src/app/views", {

    express:server,

    autoescape:false,

    noCache: true

})

server.listen(5000, function(){

*console*.log("server is running")

})

PASTA RAIZ

package.json

{

  "name": "modulo05",

  "version": "1.0.0",

  "description": "",

  "main": "server.js",

  "scripts": {

    "start": "npm-run-all -p nodemon browsersync",

    "nodemon": "nodemon src/server.js",

    "browsersync": "browser-sync start --proxy http://localhost:5000 --files 'public, src/app/views'"

  },

  "dependencies": {

    "express": "^4.17.1",

    "method-override": "^3.0.0",

    "nunjucks": "^3.2.0",

    "pg": "^7.18.2"

  },

  "devDependencies": {

    "browser-sync": "^2.26.7",

    "nodemon": "^2.0.2",

    "npm-run-all": "^4.1.5"

  }

}