

## CAPÍTULO 8 – Autenticação, Área Exclusiva, Cadastro e Transações

Neste capítulo, continuaremos o desenvolvimento da aplicação, focando em recursos avançados, como autenticação, áreas exclusivas para usuários autenticados, cadastro de informações e a realização de transações na API.

- ✓ Neste capítulo, você aprenderá sobre:
- ✓ Implementação de autenticação em uma aplicação Node.js.
- ✓ Criação de áreas exclusivas acessíveis somente para usuários autenticados.
- ✓ Desenvolvimento de funcionalidades de cadastro de informações.
- ✓ Realização de transações utilizando a API.

### **Atividade 1 – Autenticação utilizando bearer Token JWT**

Essa atividade tem como objetivo:

- Realizar requisições HTTP com verbo POST usando a biblioteca axios;
- Gerar um token de autenticação JWT e armazená-lo em um cookie e no session storage do navegador;
- Enviar o token para acessar rotas protegidas da API.

Comandos:

- **npm init -y | npm install** express ejs cors cookie-parser

### **Aplicação Login e Autenticação**

Vamos ver a criação de uma aplicação que envia a combinação de usuário e senha para a API e recebe e armazena o token ou trata um erro exibindo uma mensagem.

1. Dentro do diretório "**Cap08**", crie um subdiretório com o nome "**at01-Login\_e\_Autenticação**" e abra com o VSCode.;
2. Realize o procedimento para inicializar o projeto e instalar os módulos necessários com os comandos no console:
  - ✓ **npm init -y**
  - ✓ **npm install** express ejs consign axios cors cookie-parserInsira também a entrada "**start**": "nodemon index.js" (arquivo package.json vide exemplos anteriores).
3. Crie o arquivo "**index.js**" e insira o seguinte código:

```

const express = require('express')
const app = express()
const cors = require('cors')
app.use(cors())
const cookieParser = require('cookie-parser')
app.use(cookieParser())
const consign=require('consign')
const requests = require('./controllers/requests')
var porta = '3000'
global.urlServer = 'http://localhost:3200'
app.use(express.urlencoded({extended:true}))
app.use(express.json())

// Configura o Express p/ usar o EJS como View engine
app.set('view engine','ejs')

//Define diretório para arquivos estaticos(css, imagens, js(front-end))
app.use(express.static('public'))

consign()
  .include('/controllers/rotas')
  .into(app)

app.get(`/`, async (req, res)=>{
  try {
    dados = await requests.obterPets(`pets`)
    res.status(200)
    res.render('index', {dados:dados})
  } catch (error) {
    console.log(error)
    res.status(400)
    res.render('index')
  }
})

app.listen(porta, ()=>console.log(`Servidor rodando em:
http://localhost:${porta}`))

```

4. Crie um diretório “**controllers**” e dentro dele crie um arquivo chamado “**requests.js**”, onde vamos escrever e exportar método especializados para realizar requisições HTTP assíncronas utilizando os verbos: GET, POST, PUT e DELETE.
5. O arquivo “**requests.js**” possui métodos para realizar operações com a API, especialmente relacionados à autenticação e manipulação de cookies. Insira o seguinte código:

```

const axios = require('axios')
module.exports={
  obterPets: async (rota)=>{

    let uri = `${urlServer}/${rota}`
    let dados = await axios.get(uri)
    return [...dados.data]
  },
  realizarLogin: async (req, rota)=>{
    let uri = `${urlServer}/${rota}`
    let resp = await axios.post(uri, {...req.body})
    return resp.data
  },
  gravarCookie:(res, token)=>{
    res.cookie('Authorization', token, {
      //httpOnly: true,
      secure: true,
      sameSite: 'strict',
      //expires: new Date(Date.now() + 60 * 60 * 1000), //+1 hora com
data/hra definida
      maxAge: 60 * 60 * 1000 //+1 hora em milesegundos
    })
  },
  excluirCookie:(res)=>{
    //res.cookie('Authorization', 'undefined', {maxAge: 60 * 60 *
100000})
    res.cookie('Authorization', 'undefined', { expires: new Date(0) })
  }
}

```

6. Agora vamos escrever as rotas que irão utilizar os métodos. Dentro do diretório “**controllers**” crie no subdiretório “**rotas**” um arquivo chamado “**login.js**”, neste arquivo vamos implementar o código responsável por carregar o template de login, a rota post responsável por fazer a autenticação com a API e a rota de logoff. Insira o seguinte código no arquivo:

```

const requests = require('../requests')
module.exports =(app)=>{
  app.get(`/login`, async (req,
res)=>res.render('login',{dados:{message:false}}))
  app.post('/login', async (req, res)=>{
    try {
      let dados = await requests.realizarLogin(req, 'login')
      if (dados.autenticado){
        requests.gravarCookie(res, dados.token)
        res.render('area_exclusiva', {dados})
      } else{
        dados.status=401
        res.render('login', {dados})
      }
    } catch (error) {
      let dados={message:"Login Inválido!", status:401, erro:error}
      res.render('login', {dados})
    }
  })
  app.get('/logoff', async (req, res) => {
    try {
      console.log('/logoff')
      requests.excluirCookie(res)
      res.render('index')
    } catch (error) {
      console.log('Erro:', error)
    }
  })
}

```

7. Crie um segundo arquivo no subdiretório “rotas” com o nome **“area-exclusiva.js”** que abrigará a rota responsável por verificar se o acesso está autenticado e carregar o template da área exclusiva. Insira o seguinte código:

```

module.exports =(app)=>{
  app.get('/area-exclusiva', (req, res)=>{
    //Resgata o token à partir do cookie
    const token = req.headers.cookie.split('=')[1];
    if (token!==undefined){
      dados.autenticado=true
      dados.token=token
      res.render('area_exclusiva', {dados})
    }
  })
}

```

8. Para facilitar o desenvolvimento, copie as pastas “**public**” e “**views**” do último exercício do **capítulo 7(at04-ConsumindoDadosAPI)** para o diretório raiz do projeto.
9. Agora, no diretório “**views**” crie um arquivo com o nome “**area\_exclusiva.ejs**” e insira o seguinte código:

```
<%- include('partials/head.ejs') %>
<% if (typeof dados !== 'undefined') { %>
  <% if (dados.autenticado !== 'undefined' && dados.token !== 'undefined') { %>
    <script>
      //Insere o bearer token no sessionStorage
      const token = '<%= dados.token %>'
      sessionStorage.setItem('Authorization', `Bearer ${token}`)
    </script>
  <% } %>
<% } %>
<title>Área Restrita - Amigo do Pet</title>
</head>
<body>
  <div class="container-fluid">
    <!-- navbar -->
    <%- include('partials/menu.ejs') %>
  </div>
  <br>
  <!-- Formulario Login -->
  <main>
    <div class="card mt-5" id="panel-tab">
      <div class="mb-3 card-body">
        <div class="bg-light m-auto w-100 d-block">
          <ul class="nav nav-tabs" id="myTab" role="tablist">
            <li class="nav-item" role="presentation">
              <button class="nav-link active" id="perfil-tab"
data-bs-toggle="tab" data-bs-target="#perfil-pane" type="button" role="tab"
aria-controls="perfil-pane" aria-selected="true">
                <small class="fs-6">Meu Perfil</small></button>
            </li>
            <li class="nav-item" role="presentation">
              <button class="nav-link" id="pets-tab" data-bs-
toggle="tab" data-bs-target="#pets-pane" type="button" role="tab" aria-
controls="pets-pane" aria-selected="false">
                <small class="fs-6">Meus Pets</small></button>
            </li>
            <li class="nav-item" role="presentation">
              <button class="nav-link" id="interesse-tab" data-bs-
toggle="tab" data-bs-target="#interesse-pane" type="button" role="tab" aria-
controls="interesse-pane" aria-selected="false">
                <small class="fs-6">Meus Interesses</small></button>
            </li>
          </ul>
          <div class="tab-content" id="myTabContent">
```

```

        <div class="tab-pane fade show active" id="perfil-pane"
role="tabpanel" aria-labelledby="perfil-tab" tabindex="0">
            <p>Perfil</p>
        </div>
        <div class="tab-pane fade show" id="pets-pane"
role="tabpanel" aria-labelledby="pets-tab" tabindex="1">
            <p>Pets</p>
        </div>
        <div class="tab-pane fade" id="interesse-pane"
role="tabpanel" aria-labelledby="interesse-tab" tabindex="2">
            <p>Interesses</p>
        </div>
    </div>
</div>
</main>
<%- include('partials/footer.ejs') %>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.
js" integrity="sha384-
MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
crossorigin="anonymous"></script>
</body>
</html>

```

10. Para ter um menu dinâmico com opções referentes à quando se está ou não autenticado, modifique a partial “**menu.js**” (no subdiretório “views/partials”) para o código a seguir:

```

<div class="container-fluid">
    <nav class="row navbar navbar-expand-lg navbar-light bg-light fixed-top
clearfix" role="navigation">
        <div class="col-sm-12 col-md-12 col-lg-2">
            <div class="row">
                <div class="col-sm-6 col-md-10 col-lg-12 mx-auto" style="max-
width: 200px;">
                    <a class="navbar-brand" href="/">
                        
                    </a>
                </div>
                <div class="col-sm-2 col-md-2 col-lg-0 my-auto py-2"
style="max-width: 100px;">
                    <div class="mx-auto">
                        <button id="btnToggleMenu" class="navbar-toggler"
type="button"
data-bs-toggle="collapse" data-bs-
target="#menuColapse"

```

```

        aria-controls="menuColapse" aria-expanded="true"
aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
    </button>
</div>
</div>
</div>
</div>
<div class="col-sm-12 col-md-12 col-lg-10">
    <div class="container">
        <div class="collapse navbar-collapse" id="menuColapse">
            <ul id="listaMenu" class="navbar-nav me-auto mb-2 mb-lg-
0">
                <li class="nav-item">
                    <a href="/" class="nav-link clearfix fs-5 text-
warning" onclick="toggleMenu()">HOME</a>
                </li>
            </ul>
            <ul class="navbar-nav mb-2 mb-lg-0">
                <li class="nav-item">
                    <a id="menuLogin" class="nav-link fs-5 text-
warning" aria-current="page" onclick="excluiToken()"></img></a>
                </li>
            </ul>
        </div>
    </div>
</div>
</nav>
<hr>
</div>
<script>
    let menuLogin=document.getElementById('menuLogin')
    let listaMenu=document.getElementById('listaMenu')
    let logado=false
    if (sessionStorage.getItem('Authorization')){
        menuLogin.innerHTML='Sair </img>'
        logado=true
        menuLogin.href='/'
        let li = document.createElement('li')
        li.classList.add('nav-item')
        let a = document.createElement('a')
        a.classList.add('nav-link', 'clearfix', 'fs-5', 'text-warning')
        a.href='/area-exclusiva'
        a.innerHTML = 'AREA EXCLUSIVA'
        li.appendChild(a)
        listaMenu.appendChild(li)
    } else {

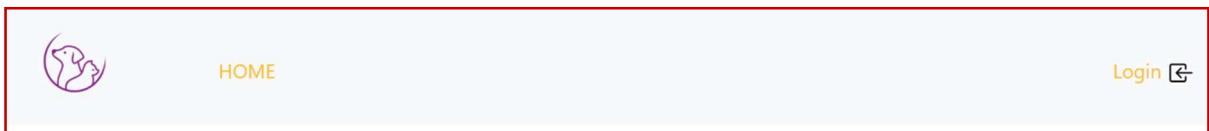
```

```

        menuLogin.innerHTML='Login'
        menuLogin.href='/login'
    }
    var excluirtoken=()=>{
        if (logado){
            sessionStorage.removeItem('Authorization')
            document.cookie = `Authorization=; maxAge:-1; path=/;`
            logado=false
        }
    }
}
</script>

```

11. Para podermos testar este código é necessário termos o **serviço da nossa API inicializado** e “escutando” na porta 3200. Portanto, navegue até o diretório da última versão da nossa API(último exercício do **capítulo 6 - at03-APIRest\_Amigo\_do\_Pet-V\_1\_1**) e inicialize o serviço com o comando **“npm start”**.
12. Com o serviço da API rodando, agora podemos testar nosso código. Execute o comando **“npm start”** também no exercício atual. Certifique-se que tenha um usuário cadastrado no banco de dados e se lembre a senha de cadastro. Caso não se lembre, pode utilizar o postman para cadastrar um novo usuário.
13. Com as credenciais de um e-mail e senha válidos acesse a rota **/login** e realize testes de autenticação.
14. Poderemos observar a seguinte exibição nos templates:



Menu sem autenticação



Acesso

[Nova Conta](#)

Acesso ao menu que aponta para a rota **/login**



Acesso

Login

joao@teste.com

Senha

...

Enviar

Conta de email inválida.

Nova Conta

Acesso

Login

joao@teste.com

Senha


...

Enviar

Senha inválida.

Nova Conta

Tratamento de erro com dados de e-mail e/ou senha inconsistentes



HOME AREA EXCLUSIVA

Sair

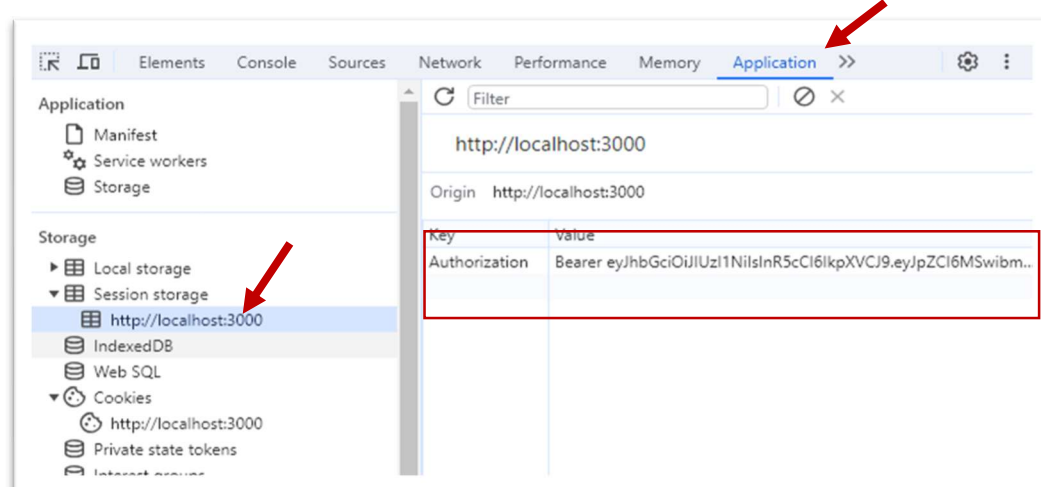
Meu Perfil

Meus Pets

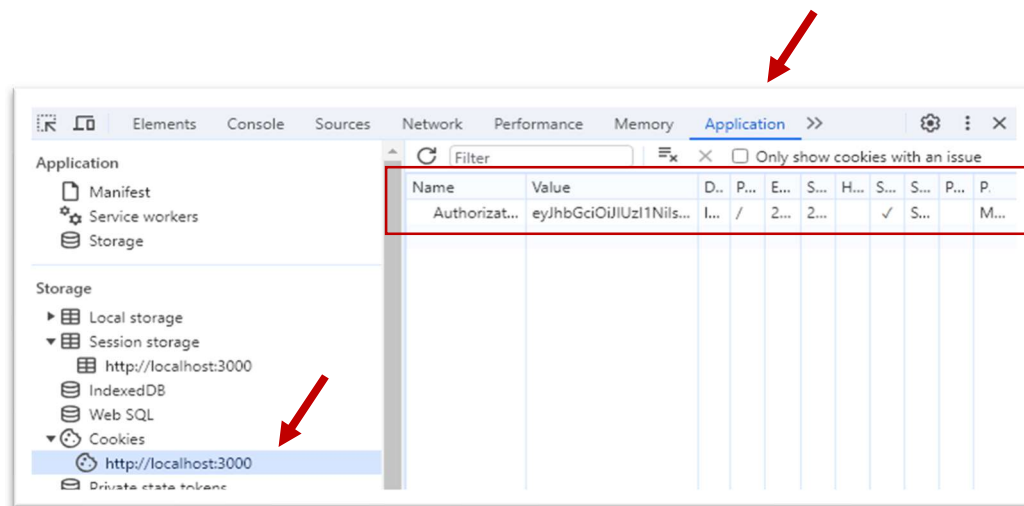
Meus Interesses

Perfil

Menu autenticado e página da área exclusiva após acertar os dados de login.



Observe o token em session storage(ferramentas de desenvolvedor no navegador com a tecla F12).



Token na área da cookie (abra as ferramentas de desenvolvedor no navegador utilizando a tecla F12).

## **Atividade 2 – Aperfeiçoamento da API para proteger rotas e atender as demandas de cadastro e transações**

Essa atividade tem como objetivo:

- Preparar a API para tratar requisições em rotas protegidas pelo token;
- Aperfeiçoar a API para atender requisições de dados e novos cadastros;

### **Aperfeiçoamento da API**

1. Copie para o diretório "Cap08" o diretório da última versão da nossa API que faz parte do capítulo6 at03-APIRest\_Amigo\_do\_Pet-V\_1\_1, renomeie a cópia para "at02-APIRest\_Amigo\_do\_Pet-V\_1\_2" e abra com o VSCode;
2. Refatore o código do "index.js" para o seguinte código:

```
const express = require('express')
const consign=require('consign')
const app = express()
const cors = require('cors')
app.use(cors({
  origin: '*', // Permite todas as origens
  credentials: true // Se necessário para permitir o envio de cookies
}))
var porta = '3200'
app.use(express.urlencoded({extended:false}))
app.use(express.json())
app.get('/', (req, res)=>res.send('API - Amigo do Pet'))
consign()
  .include('./controllers/rotas')
  .into(app)
app.listen(porta, ()=>console.log(`Servidor rodando em:
http://localhost:${porta}`))
```

3. Agora vamos iniciar o processo de **refatoração das rotas protegidas**, as linhas alteradas haverá a **indicação** de um **comentário “//alterar”**. Apenas as rotas POST “/login” e “/usuários” não devem ser **protegidas pelo middleware “auth.validarToken,”** (destacado pelo retângulo vermelho) que verifica a autenticidade do token e retorna os dados do usuário autenticado. Vamos iniciar com o código do arquivo “/controllers/rotas/ usuário.js”:

```
const model = new require('../models/usuario')
const auth = require('../auth')
const validacao = require('../validacao')
const rota = 'usuarios'
module.exports = (app)=>{
  app.get(`/${rota}`, auth.validarToken, async (req, res)=>{ //rota
    protegida
    try {
      let id=req.usuarioAtual.id //Alterar
      let dados = await model.findByPk(id)
      delete dados.dataValues.senha
      res.json(dados).status(200)
    } catch (error) {
      res.json(error).status(400)
    }
  })
  app.post(`/${rota}`, async (req, res)=>{
    try {
      let dados = req.body
      let dadosLogin = await validacao.validarCadastro(dados, model)
      if (dadosLogin.validacao){
        dados.senha = await auth.criptografarSenha(dados.senha)
        let respBd = await model.create(dados)
        delete respBd.dataValues.senha
        res.json(respBd).status(201)
      } else {
        res.json(dadosLogin).status(200)
      }
    } catch (error) {
      res.json(error).status(422)
    }
  })
  app.put(`/${rota}`, auth.validarToken, async (req, res) => {
    try {
      let id = req.usuarioAtual.id //alterar
      let {nome, cpf, telefone, whatsapp} = req.body
      //Atualizar dados de login exigem regras específicas de validação
      let dados = {nome:nome, cpf:cpf, telefone:telefone,
        whatsapp:whatsapp}
      let respBd = await model.update(dados, {where:{id:id}})
      res.json(respBd).status(200)
    }
  })
}
```

```

    } catch (error) {
      res.json(error).status(400)
    }
  })
  app.delete(`/${rota}`, auth.validarToken, async (req, res) => {
    try {
      let id = req.usuarioAtual.id //alterar
      //A exclusão de registro exigem regras específicas de validação
      além deste escopo
      let respBd = await model.destroy({where:{id:id}})
      res.json(respBd)
    } catch (error) {
      res.json(error).status(400)
    }
  })
}

```

Vale observar que as rotas PUT e DELETE que utilizavam parâmetros na URL, deixou de utiliza-los e passaram a utilizar o "id" do usuários que foram extraídos do token, está é uma boa prática.

4. Agora vamos refatorar o código do arquivo do diretório **"/controllers/rotas/pets.js"**, o mesmo padrão será seguido para os próximos passos:

```

const model = new require('../models/pet')
const usuario = new require('../models/usuario')
const auth = require('../auth')
const rota = 'pets'
module.exports = (app)=>{
  app.get(`/${rota}/:id?`, async (req, res)=>{
    try {
      let dados = req.params.id? await
      model.findOne({where:{id:req.params.id}}) : //Alterar
      await model.findAll({include:[{model:usuario}], {raw: true,
      order:[['id', 'DESC']]})
      res.json(dados).status(200)
    } catch (error) {
      res.json(error).status(400)
    }
  })
  //Criar esta rota
  app.get(`/${rota}`, auth.validarToken, async (req, res) => {
    try {
      let id=parseInt(req.usuarioAtual.id)
      let dados = await model.findAll({where:{usuarioId:id}}, {raw:
      true, order:[['id', 'DESC']]})
      res.json(dados).status(200)
    } catch (error) {
      res.json(error).status(400)
    }
  })
}

```

```

    }
  )),
  app.post(`/${rota}`, auth.validarToken, async (req, res)=>{
    try {
      let dados = req.body
      dados.usuarioId=req.usuarioAtual.id //Alterar
      let respBd = await model.create(dados)
      res.json(respBd).status(200)
    } catch (error) {
      res.json(error).status(400)
    }
  })
  app.put(`/${rota}`, auth.validarToken, async (req, res) => {
    try {
      let id = req.usuarioAtual.id //Alterar
      let dados = req.body
      console.log(dados)
      let respBd = await model.update(dados, {where:{id:id}})
      res.json(respBd)
    } catch (error) {
      res.json(error).status(400)
    }
  })
  app.delete(`/${rota}`, auth.validarToken, async (req, res) => {
    try {
      let id = req.usuarioAtual.id //Alterar
      let respBd = await model.destroy({where:{id:id}})
      res.json(respBd)
    } catch (error) {
      res.json(error).status(400)
    }
  })
}

```

5. Agora vamos refatorar o código do arquivo do diretório **“/controllers/rotas/ doacao.js”**:

```

const model = new require('../../models/pet')
const usuario = new require('../../models/usuario')
const auth = require('../auth')
const rota = 'pets'
module.exports = (app)=>{
  app.get(`/${rota}/:id?`, async (req, res)=>{
    try {
      let dados = req.params.id? await
model.findOne({where:{id:req.params.id}}) : //Alterar
      await model.findAll({include:[{model:usuario}], {raw: true,
order:[['id', 'DESC']]})
      res.json(dados).status(200)
    }
  })
}

```

```

        } catch (error) {
            res.json(error).status(400)
        }
    })
    app.get(`/obter/${rota}`, auth.validarToken, async (req, res) => { //Criar
esta nova rota
        try {
            let id=parseInt(req.usuarioAtual.id) //Alterar
            let dados = await model.findAll({where:{usuarioId:id}}, {raw:
true, order:[['id','DESC']]})
            res.json(dados).status(200)
        } catch (error) {
            res.json(error).status(400)
        }
    }),
    app.post(`//${rota}`, auth.validarToken, async (req, res)=>{
        try {
            let dados = req.body
            dados.usuarioId=req.usuarioAtual.id //Adiciona o Id do usuario
autenticado
            let respBd = await model.create(dados)
            res.json(respBd).status(200)
        } catch (error) {
            res.json(error).status(400)
        }
    })
    app.put(`//${rota}`, auth.validarToken, async (req, res) => {
        try {
            let id = req.usuarioAtual.id //Alterar
            let dados = req.body
            let respBd = await model.update(dados, {where:{id:id}})
            res.json(respBd)
        } catch (error) {
            res.json(error).status(400)
        }
    })
    app.delete(`//${rota}`, auth.validarToken, async (req, res) => {
        try {
            let id = req.usuarioAtual.id //Alterar
            let respBd = await model.destroy({where:{id:id}})
            res.json(respBd)
        } catch (error) {
            res.json(error).status(400)
        }
    })
}

```

6. Agora vamos refatorar o código do arquivo do diretório **“/controllers/rotas/ login.js”**:

```

const model = new require('../models/usuario')
const auth = require('../auth')
const validacao = require('../validacao')

module.exports = (app) => {
  app.post(`/login`, async (req, res) => {
    try {
      let dados = req.body
      let validaLogin = await validacao.validarLogin(dados, model)
      if (validaLogin.autenticado) { //Verifica se email e senha são
        consistentes
          let {id, nome, email} = validaLogin.usuario.dataValues
          dados = {id, nome, email} //Desestruturação dos dados
          validados
          let token = await auth.gerarToken(dados) //Gera um token
          JWT
          return res.json({dados, autenticado:true,
            token:token}).status(200)
        } else {
          return res.json(validaLogin).status(401)
        }
      } catch (error) {
        return res.json(error).status(400)
      }
    })
  })
}

```

7. Após concluir as alterações, rode o código e faça testes de funcionalidades utilizando o postman para verificar possíveis inconsistências e erros de digitação.

### **Atividade 3 – Aperfeiçoamento do módulo requests.js**

Essa atividade tem como objetivo:

- Criar métodos que realizar requisições HTTP usando a biblioteca Axios que aceitam argumentos dinâmicos que podem lidar com requisições protegidas ou não;

#### **Aperfeiçoamento do módulo requests**

1. Copie para o diretório do exercício “at01-Login\_e\_Autenticação” do capítulo 8, renomeie a cópia para “at03-Metodos\_Especialistas\_HTTP” e abra com o VSCode;
2. Refatore o código do arquivo “requests.js” que se encontra no diretório “controllers”, para o seguinte código:

```

const axios = require('axios')
module.exports={
  requisicaoGet:async (...dataReq) => {
    try {
      dataReq[0]=`${urlServer}/${dataReq[0]}`
      if (dataReq[1]!==undefined){
        let config = {
          headers: {
            'Authorization':dataReq[1] // Adiciona o cabeçalho de
            autorização com o token
          }
        }
        dataReq[1]= config
      }
      let resp = await axios.get(...dataReq)
      return resp.data
    } catch (error) {
      return {status:400, message:'A requisição não pode ser
      respondida!', erro:error}
    }
  },
  requisicaoPost:async (...dataReq) => {
    try {
      dataReq[0]=`${urlServer}/${dataReq[0]}`
      if (dataReq[2]!==undefined){
        let config = {
          headers: {
            'Authorization':dataReq[2] // Adiciona o cabeçalho de
            autorização com o token
          }
        }
        dataReq[2]= config
      }
      let resp = await axios.post(...dataReq)
      return resp.data
    } catch (error) {
      return {status:400, message:'A requisição não pode ser
      respondida!', erro:error}
    }
  },
  requisicaoPut:async (rota, dados, token) => {
    try {
      let uri=`${urlServer}/${rota}`
      let config = {
        headers: {
          'Authorization':token // Adiciona o cabeçalho de
          autorização com o token
        }
      }
    }
  }
}

```



```

        let resp = await axios.put(uri, dados, config)
        return resp.data
      } catch (error) {
        return {status:400, message:'A requisição não pode ser
respondida!', erro:error}
      }
    },
    requisicaoDelete:async (rota, token) => {
      try {
        let uri=`${urlServer}/${rota}`
        let config = {
          headers: {
            'Authorization':token // Adiciona o cabeçalho de
autorização com o token
          }
        }
        let resp = await axios.delete(uri, config)
        return resp.data
      } catch (error) {
        return {status:400, message:'A requisição não pode ser
respondida!', erro:error}
      }
    },
    gravarCookie:(res, token)=>{
      res.cookie('Authorization', token, {
        //httpOnly: true,
        secure: true,
        sameSite: 'strict',
        //expires: new Date(Date.now() + 60 * 60 * 1000), //+1 hora com
data/hra definida
        maxAge: 60 * 60 * 1000 //+1 hora em milesegundos
      })
    },
    excluirCookie:(res)=>{
      //res.cookie('Authorization', 'undefined', {maxAge: 60 * 60 * 100000})
      res.cookie('Authorization', 'undefined', { expires: new Date(0) })
    }
  }
}

```

3. Vamos precisar refatorar os arquivos “**index.js**” e o arquivo “**login.js**” que fazem uso do módulo “requests.js”. Vamos iniciar pelo “index.js” substituindo a linha a seguir:

```
dados = await requests.obterPets(`pets`)
```

por:

```
dados = await requests.requisicaoGet(`pets`)
```

4. Agora é a vez do arquivo “login.js” que se encontra no diretório “/controllers/rotas”, substitua a linha:

```
let dados = await requests.realizarLogin(req, 'login')
```

por:

```
let dados = await requests.requisicaoPost('login', req.body)
```

5. Rode o código e verifique se existe alguma inconsistência no código após a refatoração. Caso exista corrija o problema antes de prosseguir para a próxima atividade.

## **Atividade 4 – Cadastrar, editar e exibir registros de usuários**

Essa atividade tem como objetivo:

- Criar editar e exibir registros na tabela usuários;

### **Criação de rotas e templates para manipular dados de usuários**

1. Copie para o diretório do exercício “at03-Metodos\_Especialistas\_HTTP” do capítulo 8, renomeie a cópia para “at04-Cadastrar\_editar\_exibir\_usuarios” e abra com o VSCode;
2. Vá até o template “login.ejs” na pasta “views” e altere a linha:

```
<a class="navbar navbar-nav navbar-item navbar-link" href="/cadastrar/usuario">Nova  
Conta</a>
```

Por:

```
<a class="navbar navbar-nav navbar-item navbar-link" href="/novo-usuario">Nova  
Conta</a>
```

3. Agora crie um arquivo chamado “usuario.js” no diretório “/contorllers/rotas” e insira o seguinte código:

```
const requests = require('../requests')  
module.exports =(app)=>{  
  app.get(`/novo-usuario`, async (req, res)=>res.render('novo_usuario')),  
  app.post('/novo-usuario', async (req, res)=>{  
    try {  
      let dados = await requests.requisicaoPost('usuarios', req.body)  
      res.render('confirmacao', dados)  
    } catch (error) {  
      let dados={message:"Não foi possível realizar o cadastro.",  
status:401, erro:error}  
      res.render('login', dados)  
    }  
  })  
}
```

```

    }
  )),
  app.post('/editar-perfil', async(req, res) => {
    let dados={}
    if (req.headers.cookie==='Authorization=' || req.headers.cookie===''){
      dados.autenticado=false
      dados.token=undefined
      res.render('login',{dados:{message:'Sessão expirou. Faça
login!'}})
    } else {
      const token = req.headers.cookie.split('=')[1]
      if (token!==undefined){
        dados.autenticado=true
        dados.token=token

        //Realiza a atualização na API
        let respApi = await requests.requisicaoPut('usuarios',
req.body, `Bearer ${dados.token}`)
        dados.message=respApi>0?"Dados atualizados com sucesso!":"Os
dados não puderam ser gravados."
        dados.alert=respApi>0?"success":"danger"

        //Atualizar dados e recarregar página
        dados.meusDados = await requests.requisicaoGet(`usuarios`,
`Bearer ${dados.token}`)
        dados.meusPets = await requests.requisicaoGet(`obter/pets`,
`Bearer ${dados.token}`)
        meusInteresses = await requests.requisicaoGet(`doacoes`,
`Bearer ${dados.token}`)
        dados.meusInteresses = meusInteresses.map(m=>m.pet)

        res.render('area_exclusiva', {dados})
        //res.redirect('/area-exclusiva')
      }
    }
  })
}

```

4. Agora vamos criar um formulário de cadastro para usuários. Crie um arquivo na pasta “views” com o nome de “**novo\_usuario.ejs**” e insira o seguinte código:

```

<%- include('partials/head.ejs') %>

<title>Novo Usuário</title>
</head>
<body>
  <div class="container"></div><!-- main container-fluid -->
  <!--navBar-->
  <%- include('partials/menu.ejs') %>

```

```

    <!-- Inicio do conteúdo -->
    <div class="row card bg-light mt-2">
        <div class="col-12 card-body py-2 mx-auto bg-warning">
            <hr>
            <h2 class="display-2 text-align-justify text-center text-
break text-light">Novo Usuário</h2>
            <hr>
        </div>
        <div class="pt-3 container"><!--Div de Itens - Conteúdo-->
            <div class="row px-5"> <!-- Linha de conteúdo -->

                <div class="col-12 mb-3"> <!-- 1º item de conteúdo -->
                    <div class="card shadow rounded">
                        <div class="card-body">
                            <form name="formClientes"
id="formClientes" method="post" action="/novo-usuario">
                                <!-- Dados pessoais -->
                                <%- include('partials/cadastro_usuario')%>
                                <br><hr><br>
                            </div>
                            <div class="card-footer">
                                <div class="row">
                                    <div class="col-12"></div>
                                    <div class="col-2 d-flex justify-
content-end ms-auto me-0">
                                        <button class="btn btn-primary
form-control">Gravar</button>
                                    </div>
                                </div>
                            </form>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
<!-- Rodapé-->
<%- include('partials/footer.ejs') %>

    <script>
        var frm = document.forms
        //Tabular com Enter
        function tabularComEnter(e) {
            if (e.keyCode == 13) {
                if ((e.target.type != 'submit') ||
(e.target.tagName != 'BUTTON')) {
                    e.preventDefault();
                }
            }
        }
    </script>

```

```

        let foco = document.activeElement;
        for (let i in frm) {
            for (let j in frm[i]) {
                if (frm[i][j] === foco) {
                    frm[i][parseInt(j) + 1].focus()
                    break;
                }
            }
        }
    }

    function formatarCPF(cpf) {
        //mais formatos de validação
https://aurelio.net/regex/html5/pattern.html
        const elementoAlvo = cpf
        const cpfAtual = cpf.value
        let cpfAtualizado
        // xxx.xxx.xxx-xx
        cpfAtualizado = cpfAtual.replace(
            /(\d{3})(\d{3})(\d{3})(\d{2})/,
            function (regex, argumento1, argumento2,
argumento3, argumento4) {
                return argumento1 + '.' + argumento2 + '.' +
argumento3 + '-' + argumento4
            }
        );
        elementoAlvo.value = cpfAtualizado;
    }
    function copiarTelefone(){
        console.log(frm.telefone)
        console.log(frm.whatsapp)
        frm[0].whatsapp.value = frm[0].telefone.value
        frm[0].whatsapp.focus()
    }
</script>
</body>
</html>

```

5. Vamos criar a partial com os campos do formulário para cadastrar novo usuário. Crie um arquivo com o nome **"cadastro\_usuario"** no diretório **"views/partial"** com o seguinte código:

```

<div class="row align-items-center">
  <div class="col-5 ms-5 my-3 ps-5">
    <label for="nome" class="form-label">Nome</label>
    <input type="text" id="nome" name="nome" required="required"
      minlength="4" placeholder="Ex.: João da Silva" autofocus

```

```

        class="form-control" />
    </div>

    <div class="me-5 col-3">
        <label for="cpf" class="form-label">CPF</label>
        <!--Forçar usuario digitar no formato pattern="\d{3}\.\d{3}\.\d{3}-\d{2} somente números [0-9]{11}"-->
        <input type="text" id="cpf" name="cpf" required="required"
            pattern="\d{3}\.?\d{3}\.?\d{3}-?\d{2}" placeholder="Somente
Números"
            class="form-control" onblur="formatarCPF(this)" />
    </div>

    <!--pattern="pattern="[0-9]{2}\/[0-9]{2}\/[0-9]{4}$" min="2012-01-01"
max="2014-02-18"-->
    <div class="my-5 col-4 ms-5 my-3 ps-5">
        <label class="fom-label" for="telefone">Celular</label>
        <input type="text" id="telefone" name="telefone" required="required"
class="form-control" />
    </div>
    <div class="me-5 my-5 col-4">
        <label class="fom-label" for="whatsapp">Whatsapp</label>
        <input type="text" id="whatsapp" name="whatsapp" class="form-control"
onfocus="copiarTelefone()" />
    </div>
    <br><hr><br>
    <div class="row mx-5 px-5">
        <div class="mx-5 col-8 px-5">
            <label class="fom-label" for="email">E-mail</label>
            <input type="email" id="email" name="email" class="form-control"
required="required"/>
        </div>
    </div>

    <div class="row mx-5 px-5">
        <div class="col-4 ms-5 my-3 ps-5">
            <label class="fom-label" for="senha">Senha</label>
            <input type="password" id="senha" name="senha" class="form-
control" required="required"/>
        </div>
        <div class="col-4 me-5 my-3 pe-5">
            <label class="fom-label" for="confirmacao">Confirmação da
senha</label>
            <input type="password" id="confirmacao" name="confirmacao"
class="form-control" required="required"/>
        </div>
    </div>
</div>

```

6. Agora vamos criar um template para confirmar o cadastro para usuários. Crie um arquivo na pasta "views" com o nome de "confirmacao.ejs" e insira o seguinte código:

```
<%- include('partials/head.ejs') %>

<title>Registro</title>
</head>
<body>
  <div class="container-fluid mb-5">
    <!-- navbar -->
    <%- include('partials/menu.ejs') %>
  </div>
  <hr>
  <!-- Formulario Login -->
  <main>
    <div class="container login my-5 py-5">
      <div class="row justify-content-center">
        <div class="col-4 alert alert-success" role="alert">
          <div class="card shadow rounded">
            <div class="text-center">
              <strong class="fs-2 text-center text-
roxo">Registro Efetuado!</strong>
            </div>
            <div class="card-body navddg">
              <h2 class="text-success fs-4 badge text-wrap">
Registro realizado com sucesso!</h2>
              <a class="navbar navbar-nav nav-item nav-link
text-info" href="/login"><strong class="fs-5">Login</strong></a>
            </div>
          </div>
        </div>
      </div>
    </div>
    <% if (typeof dados !== 'undefined' && dados.message ){%>
      <div class="alert alert-danger" role="alert">
        <%= dados.message %>
      </div>
    <% } %>
  </main>

  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min
.js" integrity="sha384-
MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
crossorigin="anonymous"></script>
</body>
</html>
```

7. Neste momento já é possível realizar o cadastro de um novo usuário e realizar login na aplicação. Pode-se realizar este teste.
8. Após o teste podemos perceber a que a área exclusiva foi carregada, porém, nenhum dado foi carregado. Vamos agora criar a exibição da aba “meus dados” da área exclusiva. Para isso crie um arquivo no diretório “/views/partial” com o nome “perfil.ejs” e insira o seguinte código:

```
<div class="row d-flex justify-content-center align-items-center vh-80 pb-1
mb-1 ">
  <div class="col-md-9 col-sm-12 mx-5 px-5 py-5 mb-3">
    <div class="card shadow rounded px-5 mx-5">
      <div class="text-center mt-5 px-5">
        <% if (typeof dados !== 'undefined' && dados.message ){%>
          <div class="alert alert-<%= dados.alert %>" role="alert">
            <%= dados.message %>
          </div>
        <% } %>
        <strong class="display-6 text-center text-roxo py-5"> Meu
Perfil
          <button onclick="habilitarForm()" class="btn">
            
          </button>
        </strong>
      </div>
      <hr>
      <div class="card-body px-5 py-3">
        <div class="justify-content-center mx-5">
          <form method="post" action="/editar-perfil" >
            <div class="row align-items-center">
              <div class="col-10 ms-5 my-3">
                <label for="nome" class="form-
label">Nome</label>
                <input type="text" id="nome" name="nome"
required="required" minlength="4" class="form-control" value="<%=
dados.meusDados.nome %>" readonly/>
              </div>
              <div class="ms-5 col-10">
                <label for="cpf" class="form-
label">CPF</label>
                <!--Forçar usuario digitar no formato
pattern="\d{3}\.\d{3}\.\d{3}-\d{2} somente números [0-9]{11}"-->
                <input type="text" id="cpf" name="cpf"
required="required" pattern="\d{3}\.\d{3}\.\d{3}-\d{2}"
                class="form-control" value="<%=
dados.meusDados.cpf %>" readonly/>
              </div>
            </div>
          </form>
        </div>
      </div>
    </div>
  </div>
</div>
```



```

<label class="fom-label" for="email">E-
mail</label>

<input type="email" id="email"
name="email" class="form-control" required="required" value="<%=
dados.meusDados.email %>" readonly />
</div>
<!--pattern="pattern="[0-9]{2}\/[0-9]{2}\/[0-
9]{4}$" min="2012-01-01" max="2014-02-18"-->
<div class="ms-5 my-5 col-5 my-3">
<label class="fom-label"
for="telefone">Celular</label>
<input type="text" id="telefone"
name="telefone" required="required" class="form-control" value="<%=
dados.meusDados.telefone %>" readonly/>
</div>
<div class=" my-5 col-5">
<label class="fom-label"
for="whatsapp">Whatsapp</label>
<input type="text" id="whatsapp"
name="whatsapp" class="form-control" value="<%= dados.meusDados.whatsapp
%>" readonly/>
</div>
</div>
<!-- input type="hidden" name="Authorization"
class="d-none" -->
<hr>
<div class="col-12 d-flex justify-content-end">
<input id="btnGravar" type="submit" class="btn d-
none btn-roxo py-3 fs-5" value="Gravar" />
</div>
</form>
</div>

</div>
</div>
</div>
<script>
var frm = document.forms[0]
var edicao=true
//frm.Authorization.value=sessionStorage.getItem('Authorization')
function habilitarForm(){
edicao=!edicao? true:false
frm.nome.readOnly=edicao
frm.cpf.readOnly=edicao
//frm.email.readOnly=edicao
frm.telefone.readOnly=edicao

```

```

    frm.whatsapp.readOnly=edicao
    frm.nome.focus()
    frm.btnGravar.classList.add('d-flex')
    frm.btnGravar.classList.toggle('d-none')
  }
</script>

```

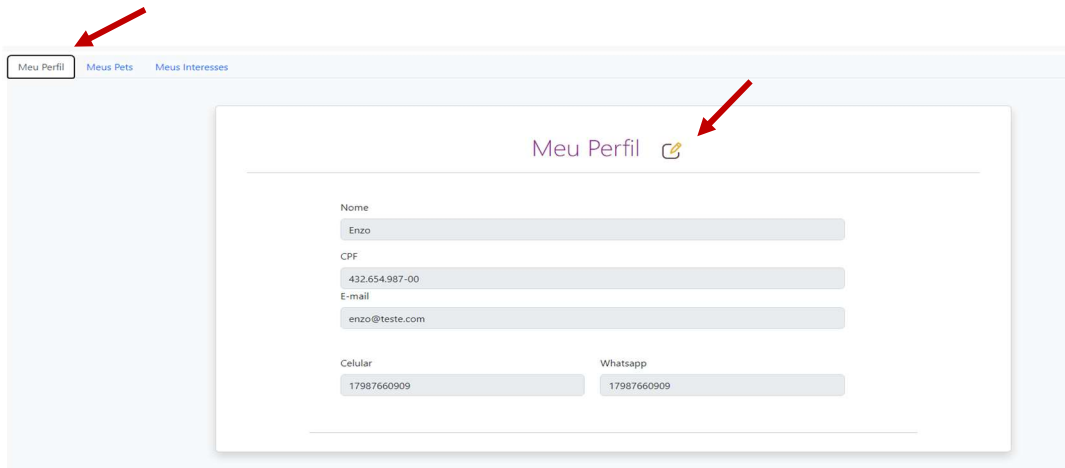
9. Refatore o arquivo **“area-exclusiva.js”** do diretório **“/controllers/rotas”** para o seguinte código:

```

const requests = require('../requests')
module.exports =(app)=>{
  app.get('/area-exclusiva', async (req, res)=>{
    try {
      let dados={}
      if (req.headers.cookie==='Authorization=' || req.headers.cookie===''){
        dados.autenticado=false
        dados.token=undefined
        res.render('login',{dados:{message:'Sessão expirou. Faça login!'}})
      } else {
        const token = req.headers.cookie.split('=')[1]
        if (token!==undefined){
          dados.autenticado=true
          dados.token=token
          dados.meusDados = await requests.requisicaoGet(`usuarios`,
`Bearer ${token}`)
          dados.meusPets = await requests.requisicaoGet(`obter/pets`,
`Bearer ${token}`)
          dados.meusInteresses = await requests.requisicaoGet(`doacoes`,
`Bearer ${token}`)
          dados.meusInteresses = await
requests.requisicaoGet(`doacoes`, `Bearer ${token}`)
          meusInteresses = await requests.requisicaoGet(`doacoes`,
`Bearer ${token}`)
          dados.meusInteresses = meusInteresses.map(m=>m.pet)
          res.render('area_exclusiva', {dados})
        }
      }
    } catch (error) {
      res.render('login',{dados:{message:'Sessão expirou. Faça login!'}})
    }
  })
}

```

10. A partir de agora os dados do usuário serão exibidos na área exclusiva e será possível editar o registro.



Tela Meu Perfil da área exclusiva.

## **Atividade 5 – Realizar transações de registro interesse nos Pets**

Essa atividade tem como objetivo:

- Registrar interesse nos Pets;

### **Criação de rota, modal e exibição de registros de interesse nos Pets**

1. Copie para o diretório do exercício “at04-Cadastrar\_editar\_exibir\_usuarios” do capítulo 8, renomeie a cópia para “at05-Registro\_de\_interesse” e abra com o VSCode;
2. No arquivo “area-exclusiva.js” do diretório “/controllers/rotas” acrescente a seguinte rota em “module.exports”:

```
app.post("/interesse", async (req, res)=>{
  let dados = {}
  if (req.headers.cookie==='Authorization=' || req.headers.cookie===''){
    dados.autenticado=false
    dados.token=undefined
    res.render('login',{dados:{message:'Sessão expirou. Faça login!'}})
  } else {
    const token = req.headers.cookie.split('=')[1]
    if (token!==undefined){
      dados = req.body
      dados.status = 'Andamento'
      let resp = await requests.requisicaoPost(`cadastrar/doacoes`,
dados, `Bearer ${token}`)
      dados.message=resp.message
      dados.autenticado=true
      dados.token=token
      res.json(resp)
    }
  }
})
```

3. No diretório de partials crie um arquivo com o nome “modal\_interesse.ejs” e insira o código:

```

<!-- Modal -->
<div class="modal" id="modalInteresse">
  <div class="modal-dialog">
    <div class="modal-content">

      <!-- Cabeçalho do Modal -->
      <div class="modal-header">
        <h4 class="modal-title">Registrar Interesse</h4>
        <button type="button" class="btn-close" data-bs-dismiss="modal"
aria-label="Close"></button>
      </div>

      <!-- Corpo do Modal -->
      <div class="modal-body">
        <form id="frmInteresse">
          <div class="row">
            <div class="col-4">
              <label for="data" class="form-label">Data</label>
              <input type="date" class="form-control" name="data" readonly>
            </div>
          </div>
          <br>
          <h3>Dados do Pet:</h3>

          <div class="mb-3 row">
            <input type="text" name="idPet" hidden readonly>
            <div class="col-md-4 col-sm-12">
              <label for="nomePet" class="form-label">Nome do Pet</label>
              <input type="text" class="form-control" name="nomePet"
readonly>
            </div>
            <div class="col-md-4 col-sm-6">
              <label for="idade" class="form-label">Idade</label>
              <input type="text" class="form-control" name="idade" readonly>
            </div>
            <div class="col-md-4 col-sm-6">
              <label for="raca" class="form-label">raca</label>
              <input type="text" class="form-control" name="raca" readonly>
            </div>
          </div>
          <br>
          <h3>Dados do Usuario:</h3>
          <hr>
          <div class="mb-3 row">
            <input type="text" name="idUsuario" hidden readonly>
            <div class="col-md-4 col-sm-12">
              <label for="nomeUsuario" class="form-label">Nome</label>
              <input type="text" class="form-control" name="nomeUsuario"
readonly>

```

```

        </div>
        <div class="col-md-4 col-sm-6">
            <label for="telefone" class="form-label">Telefone</label>
            <input type="text" class="form-control" name="telefone"
readonly>
        </div>
        <div class="col-md-4 col-sm-6">
            <label for="whatsapp" class="form-label">Whatsapp</label>
            <input type="text" class="form-control" name="whatsapp"
readonly>
        </div>
    </div>
</div>
</form>
<br>
<p class="fst-normal">
    Declaro ao confirmar este formulário que é de meu interesse
    adotar o Pet acima descrito e por isso
    <span class="fw-bolder"> concordo em compartilhar </span> os
    seguintes dados pessoais:
    <span class="fw-bolder"> Nome, Telefone, WahatsApp </span>

</p>
<div class="form-check">
    <input class="form-check-input" type="checkbox" value=""
id="checkDeAcordo">
    <label class="form-check-label" for="flexCheckDefault">
        Esou ciênte e concordo com o encaminhamento dos meus dados.
    </label>
</div>
</div>
<hr>
<!-- Botão de envio -->
<button type="button" class="btn btn-primary"
onclick="submeterDados()">Confirmar Interesse</button>
</div>
</div>
</div>
</div>
<script>
    // Instanciar um objeto Modal
    let divInteresse = document.getElementById('modalInteresse')
    let modalInteresse = new bootstrap.Modal(divInteresse)
    let frmInteresse=document.getElementById('frmInteresse')
    let checkDeAcordo = document.getElementById('checkDeAcordo')
    async function abrirModalInteresse(id) {
        // Abrir o modal
        if ( logado){
            let url = `${urlServidor}/pets/${id}`
            let pet = await axios.get(url)
            pet = pet.data

```

```

url = `${urlServidor}/usuarios`
let config = {
    headers: {
        'Authorization':sessionStorage.getItem('Authorization')
    }
}
// Adiciona o cabeçalho de autorização com o token

let usuario = await axios.get(url, config)
usuario = usuario.data
carregarInteresseFrm(pet, usuario)
modalInteresse.show()
} else {
    window.location.href = "/login";
}
}

function carregarInteresseFrm(pet, usuario){
    frmInteresse.data.value = new Date().toISOString().split('T')[0]
    checkDeAcordo.checked=false
    frmInteresse.idPet.value = pet.id
    frmInteresse.nomePet.value = pet.nome
    frmInteresse.idade.value = pet.idade
    frmInteresse.raca.value = pet.raca
    frmInteresse.idUsuario.value = usuario.id
    frmInteresse.nomeUsuario.value = usuario.nome
    frmInteresse.telefone.value = usuario.telefone
    frmInteresse.whatsapp.value = usuario.whatsapp
}

async function submeterDados(){

    if (checkDeAcordo.checked) {
        let dados = {petId:frmInteresse.idPet.value,
usuarioId:frmInteresse.idUsuario.value}
        let resp = await axios.post('/interesse', dados)
        fecharModalInteresse()
        alert(resp.data.message)
    } else{
        alert('Para confirmar seu interesse é necessário que confirme a
ciência do compartilhamento de dados.')
    }
}

function fecharModalInteresse() {
    // Fechar o modal
    modalInteresse.hide();
}

function postInteresse() {
    fecharModalInteresse();
}

</script>

```

4. Acrescente a linha de importação da partial anterior no arquivo do template “index.ejs” no diretório “views”, conforme orientações a seguir:

```
<%- include('partials/modal_interesse.ejs') %> <!--Inserir apenas esta
linha(próximo da linha 57), as demais são referência.-->
    <div class="pt-3 container"><!--Div de Itens - Conteúdo-->
        <div class="row"> <!-- Linha de conteúdo -->

                <% dados.forEach((pet)=>{ %>
```

5. Agora vamos exibir os dados de interesse na área exclusiva, para isso crie outro arquivo no diretório de partial como nome “meus\_interesses.ejs” e insira o seguinte código:

```
<hr>
<div class="row"> <!-- Linha de conteúdo -->
<strong class="display-6 text-center text-roxo py-5"> Meus
Interesses</strong>
<% dados.meusInteresses.forEach((pet)=>{ %>
<div class="col-md-4 col-sm-12 mb-3"> <!-- 1º item de conteúdo -->
<div class="card shadow rounded">

<div class="card-body">
<h3 class="card-title display-5 fs-2"><strong><%= pet.nome %></strong></h3>
<strong class="card-text text-break fs-4"><%= pet.especie %></strong>
<p class="card-text text-break fs-5"><strong>Idade: </strong><small><%=
pet.idade %></small></p>
<p class="card-text fs-5"><strong> Obs: </strong><small><%= pet.obs
%></small></p>
<a href="#" class="btn btn-roxo py-3 fs-5"><strong>Excluir</strong></a>
</div>
</div>
</div> <!--Fim 1º item de conteúdo -->
<% }) %>
</div> <!-- pets -->
```

6. Insira a linha de importação da partial anterior no arquivo “area-exclusiva.ejs” no diretório “views”, conforme orientação a seguir:

```
<div class="tab-pane fade show" id="interesse-pane" role="tabpanel" aria-
labelledby="interesse-tab" tabindex="2">
    <!--Interesses-->
        <%- include('partials/meus_interesses.ejs')%>
    <!--Interesses-->
</div>
```

7. A partir de agora já podem ser inseridos registros de interesse nos pets e serem exibidos na área exclusiva na aba meus interesses.

## **Atividade 6 – Cadastrar, exibir pets e lista de interessados**

Essa atividade tem como objetivo:

- Cadastrar, exibir pets e lista de interessados;

### Criação de rota, modal e exibição de registros de interesse nos Pets

1. Copie para o diretório do exercício “at05-Registro\_de\_interesse” do capítulo 8, renomeie a cópia para “at06-Cadastrar\_exibir\_Pets” e abra com o VSCode;
2. Vamos iniciar criando uma lista de “meus pets” na área exclusiva. Para isso cria no diretório de partials um arquivo com o nome “meus\_pets.ejs” e insira o seguinte código:

```
<strong class="display-6 text-center text-roxo py-5"> Meus Pets
  <button onclick="abrirModalAddPts()" class="btn">
    </img>
  </button>
</strong>
</div>
<hr>
<div class="row"> <!-- Linha de conteúdo -->
  <% dados.meusPets.forEach((pet)=>{ %>
    <div class="col-md-4 col-sm-12 mb-3"> <!-- 1º item de conteúdo -->
      <div class="card shadow rounded">
        
        <div class="card-body">
          <h3 class="card-title display-5 fs-2"><strong><%= pet.nome
%></strong></h3>
          <strong class="card-text text-break fs-4"><%= pet.especie %></strong>
          <p class="card-text text-break fs-5"><strong>Idade:
</strong><small><%= pet.idade %></small></p>
          <p class="card-text fs-5"><strong> Obs:
</strong><small><%= pet.obs %></small></p>
          <button class="btn btn-success py-3 fs-5"
onclick="abrirModalInteressados(this.value)" value="<%= pet.id
%>"><strong>Interessados</strong></button>
          <button class="btn btn-roxo py-3 fs-
5"><strong>Excluir</strong></button>
        </div>
      </div>
    </div> <!--Fim 1º item de conteúdo -->
  <% }) %>
</div> <!-- pets -->
```

3. Inclua a partil anterior no arquivo “area-exclusiva.ejs”, conforme orientação abaixo:

```
<!--pets-->
<div class="pt-3 container"><!--Div de Itens - Conteúdo-->
  <%- include('partials/meus_pets.ejs')%>
</div><!-- Fim da tab -->
```



4. Após isso os pets já podem ser exibidos na página de área exclusiva. Vamos agora avançar e construir o template para cadastrar pets. Ainda no diretório de partial crie um arquivo com o nome “**modal\_addPet.ejs**” e insira o seguinte código:

```
<!-- Modal -->
<div class="modal" id="modalFrmAdicionarPets">
  <div class="modal-dialog">
    <div class="modal-content">
      <!-- Cabeçalho do Modal -->
      <div class="modal-header">
        <h4 class="modal-title">Adicionar Pets</h4>
        <button type="button" class="btn-close" data-bs-dismiss="modal"
aria-label="Close"></button>
      </div>
      <!-- Corpo do Modal -->
      <div class="modal-body">
        <form id="frmAddPets" method="POST" action="/novo-pet">
          <hr>
          <div class="mb-3 row">
            <div class="col-md-8 col-sm-12">
              <label for="nome" class="form-label">Nome</label>
              <input type="text" class="form-control" name="nome">
            </div>
            <div class="col-md-4 col-sm-6">
              <label for="sexo" class="form-label">Sexo</label>
              <select type="text" class="form-select" name="sexo">
                <option value="M">Macho</option>
                <option value="F">Fêmea</option>
              </select>
            </div>
            <div class="col-md-4 col-sm-6">
              <label for="especie" class="form-label">Especie</label>
              <select type="text" class="form-select" name="especie">
                <option value="cão">Cão</option>
                <option value="gato">Gato</option>
                <option value="tartaruga">Tartaruga</option>
              </select>
            </div>
            <div class="col-md-4 col-sm-6">
              <label for="raca" class="form-label">raca</label>
              <input type="text" class="form-control" name="raca">
            </div>
            <div class="col-md-4 col-sm-6">
              <label for="peso" class="form-label">Peso</label>
              <input type="text" class="form-control" name="peso">
            </div>
            <div class="col-md-4 col-sm-6">
              <label for="tamanho" class="form-label">Tamanho</label>
              <input type="text" class="form-control" name="tamanho">
            </div>
          </div>
        </form>
      </div>
    </div>
  </div>
</div>
```

```

        </div>
        <div class="col-md-4 col-sm-6">
            <label for="idade" class="form-label">Idade</label>
            <input type="text" class="form-control" name="idade">
        </div>
        <div class="col-12">
            <label for="doenca" class="form-label">Doenca</label>
            <textarea class="form-control" name="doenca"
rows="2"></textarea>
        </div>
        <div class="col-12">
            <label for="obs" class="form-label">Obs.:</label>
            <textarea class="form-control" name="obs" rows="5"></textarea>
        </div>
    </div>

    <hr>
    <input type="submit" class="btn btn-primary" value="Salvar
Registro">
    </form>
</div>
</div>
</div>
</div>
</div>
<script>
    // Instanciar um objeto Modal
    let divFrmAdicionarPets = document.getElementById('modalFrmAdicionarPets')
    let modalInteresse = new bootstrap.Modal(divFrmAdicionarPets)
    let frmAddPets=document.getElementById('frmAddPets')

    async function abrirModalAddPts() {
        // Abrir o modal
        if ( logado){
            modalInteresse.show()
        } else {
            window.location.href = "/login";
        }
    }

    function fecharModalFrmAdicionarPets() {
        // Fechar o modal
        modalInteresse.hide();
    }
</script>

```

5. Ainda no diretório de partial crie outro arquivo com o nome “**modal\_verInteressados.ejs**” que exibira a lista de usuários interessados no pet. Insira o seguinte código:

```
<!-- Modal -->
```

```

<div class="modal" id="modalInteressados">
  <div class="modal-dialog modal-lg">
    <div class="modal-content">
      <!-- Cabeçalho do Modal -->
      <div class="modal-header">
        <h4 class="modal-title">Interessados no Pet</h4>
        <button type="button" class="btn-close" data-bs-dismiss="modal"
aria-label="Close"></button>
      </div>
      <!-- Corpo do Modal -->
      <div class="modal-body">
        <div class="container mt-5">
          <hr>
          <p>Lista de interessados: </p>
          <table class="table table-striped">
            <thead>
              <tr>
                <th scope="col">Data</th>
                <th scope="col">Nome</th>
                <th scope="col">Telefone</th>
                <th scope="col">WhatsApp</th>
                <th scope="col">Ação</th>
              </tr>
            </thead>
            <tbody id="tabelaCorpo">
              <!-- Linhas serão adicionadas aqui via JavaScript -->
            </tbody>
          </table>
          <hr>
        </div>
      </div>
    </div>
  </div>
</div>
<script>
  // Instanciar um objeto Modal
  let divInteressados = document.getElementById('modalInteressados')
  let modalInteressados = new bootstrap.Modal( divInteressados)
  let tabelaCorpo = document.getElementById('tabelaCorpo')
  tabelaCorpo.innerHTML = ''

  async function abrirModalInteressados(id) {
    // Abrir o modal
    if (logado){
      url = `${urlServidor}/obter/doacoes/${id}`
      let config = {
        headers: {
          'Authorization':sessionStorage.getItem('Authorization')
        }
      }
      // Adiciona o cabeçalho de autorização com o token
    }
  }

```

```

    }
  }
  let interesse = await axios.get(url, config)

  construirTabela(interesse.data)

} else {
  window.location.href = "/login"
}
}

function construirTabela(interesses){
  if (interesses.length === 0){
    alert('Ainda não existem interessados para este Pet.')
    fecharModalInteressados()
  } else {
    interesses.forEach(interesse => {
      const tr = document.createElement('tr')
      tr.innerHTML = `
        <td>${interesse.data_interesse}</td>
        <td>${interesse.usuario.nome}</td>
        <td>${interesse.usuario.telefone}</td>
        <td>${interesse.usuario.whatsapp}</td>
        <td>
          <button class="btn btn-success" value="${interesse.usuario.id}" onclick="">
            <i class="bi bi-check"></i>
          </button>
        </td>
      `
      tabelaCorpo.appendChild(tr)
      modalInteressados.show()
    })
  }
}

function fecharModalInteressados() {
  // Fechar o modal
  modalInteressados.hide();
}

</script>

```

6. Inclua as duas partials anteriores no arquivo “**area-exclusiva.ejs**” (no final uma linha antes do footer), conforme orientação a seguir:

```

</main>

<%- include('partials/modal_addPet.ejs')%>
<%- include('partials/modal_verInteressados.ejs')%>
<%- include('partials/footer.ejs') %>

```

7. Pronto, agora a aplicação será capaz de inserir novos registros de pets, exibir os pets na área exclusiva e exibir interessados.

## **Exercícios extras**

Esta atividade tem o objetivo ampliar o conhecimento no desenvolvimento de regras de negócio de uma aplicação.

1. Desenvolva novas funcionalidades para editar e excluir registros de pets.
2. E para um usuário não registrar interesse em um pet mais de uma vez.