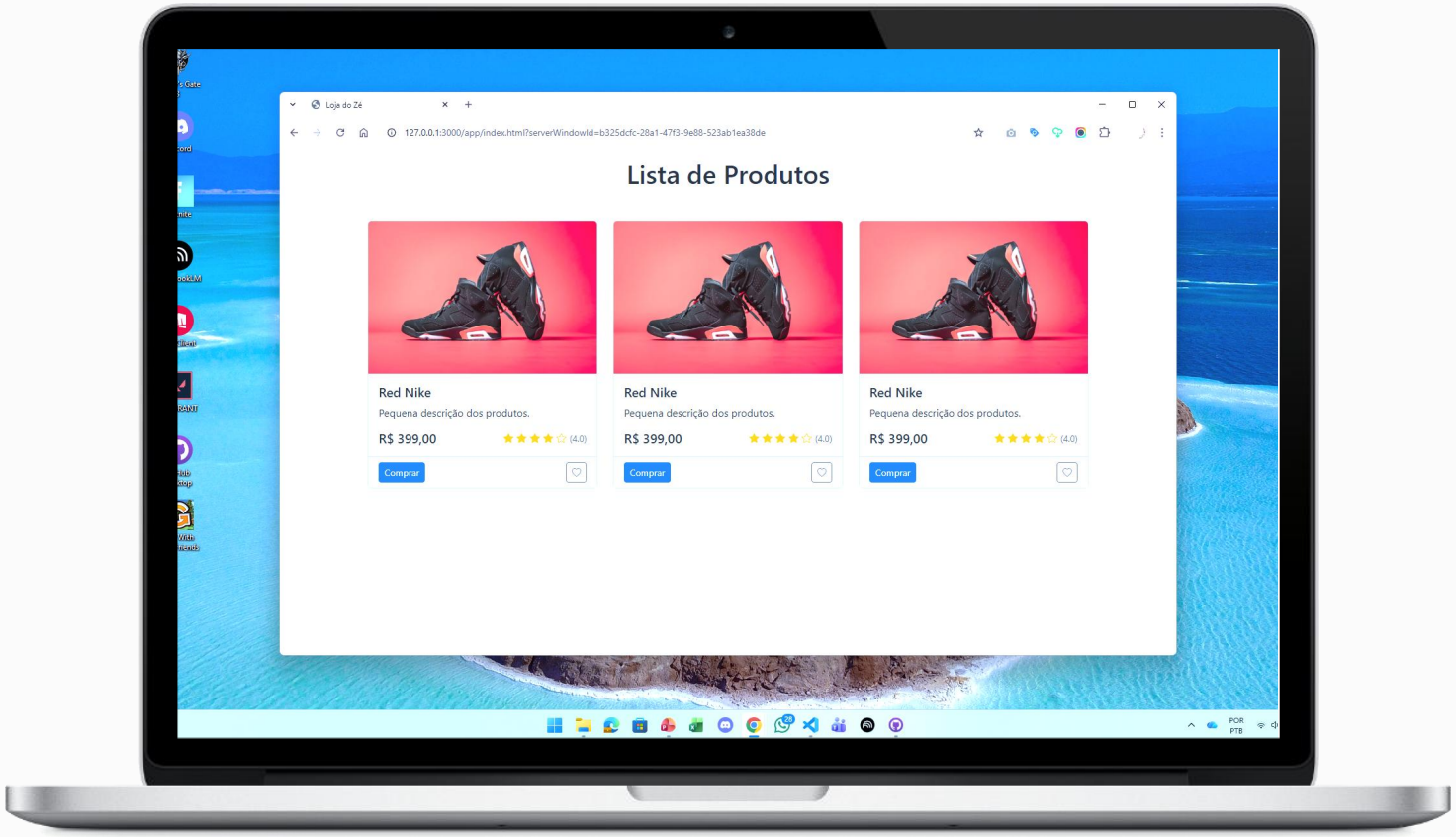
Node.js

Criação da api
Integração com FrontEnd
Conexão de dados
- array
- json
- mysql

O problema


Loja do Zé
127.0.0.1:3000/app/index.html?serverWindowId=b325dcfc-28a1-47f3-9e88-523ab1ea38de
Lista de Produtos
Red Nike
Pequena descrição dos produtos.
R$ 399,00
(4.0)
Comprar
Red Nike
Pequena descrição dos produtos.
R$ 399,00
(4.0)
Comprar
Red Nike
Pequena descrição dos produtos.
R$ 399,00
(4.0)
Comprar

Loja do Zé

127.0.0.1:3000/app/index.html?serverWindowId=b325dcfc-28a1-47f3-9e88-523ab1ea38de

Lista de Produtos

Red Nike

Pequena descrição dos produtos.

R$ 399,00 ★★★★☆ (4.0)

Comprar ♡

Red Nike

Pequena descrição dos produtos.

R$ 399,00 ★★★★☆ (4.0)

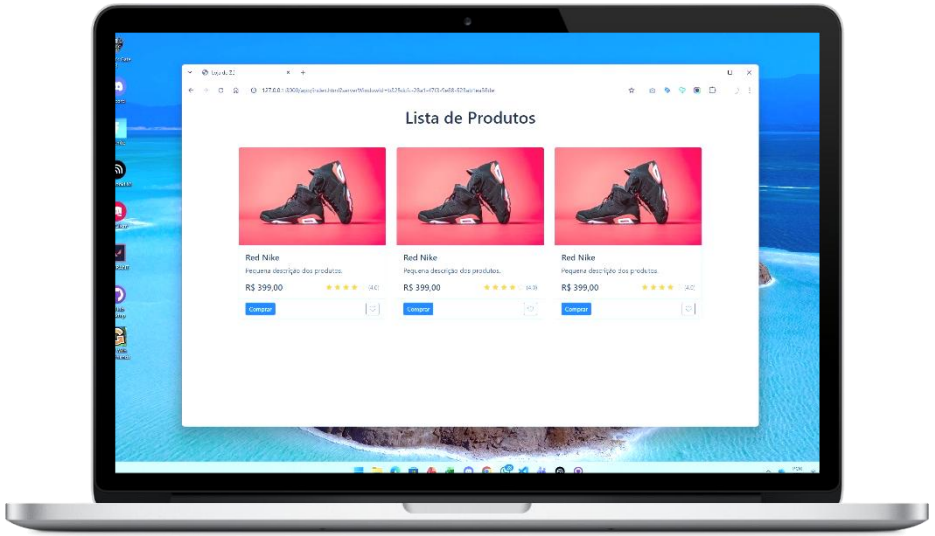
Comprar ♡

Red Nike

Pequena descrição dos produtos.

R$ 399,00 ★★★★☆ (4.0)

Comprar ♡



Definindo o problema apresentado

Backend

- criar API
  - Criar o método GET
  - Pegar os dados e enviar pro front
    - array
    - arquivo json
    - banco de dados

Conectar o frontend com o backend

- Criar a função que chama a API, recebe o retorno e monta na tela

A solução

Começando

Crie um repositório

Abra o repositório no vscode

Criar projeto – no terminal

```
# npm init
# npm install express
# npm install nodemon
```


FrontEnd
BackEnd

BACKEND

package.json

```
"scripts": {
    "start":"node index",
    "dev":"nodemon index"
}
```

index.js

```
const express = require('express')
const app = express()

/* Indica que todas as requisições podem receber Body em JSON. A partir
disso, o Express aplica um JSON.parse para o conteúdo recebido */
app.use(express.json())

app.get('/', function (req, res) {
  res.setHeader('Access-Control-Allow-Origin', '*')
  res.send('ZecaInfo')
})
app.listen(3000)
```

Teste a rota “/” da sua API

(você pode usar o navegador ou o thunderclient)

index.js - continuando

```
const lista_produtos = [
    {
        "titulo": "Red Nike",
        "foto":"foto.jpg",
        "descricao": "",
        "preco": 399.00,
        "avaliacao": 4
    }
]
```

index.js - continuando

```
// Read All - [GET] /produtos
app.get("/produtos", function (req, res) {
  res.setHeader('Access-Control-Allow-Origin', '*')
  res.send(lista_produtos)
})
```

Teste a rota “/produtos” da sua API

(você pode usar o navegador ou o thunderclient)

FRONTEND

Configurando o projeto

Na pasta do projeto crie uma pasta chamada **site-loja**

Dentro da pasta **site-loja** crie os arquivos:

- index.html
- app.js

Acesse o repositório _____ e copie o conteúdo dos arquivos index.html e app.js para os arquivos que você criou

app.js

```
function fnCarregarDados() {
    fetch('http://localhost:3000/produtos/', { method: 'GET' })
        .then(response => response.json())
        .then((produtos) => {
            produtos.forEach(produto => {
                fnMontarCardProduto(produto)
            });
        })
        .catch(erro => console.log(erro.message))
}
fnCarregarDados()
```

backend

Está tudo funcionando? Nenhum problema encontrado?

JSON

Vamos trocar os dados de um array para um JSON

index.js - continuando

Apague o array de produtos:

```
const lista_produtos = [ ...
```

No lugar coloque:

```
const lista_produtos = require('./dados.json')
```

Está tudo funcionando? Nenhum problema encontrado?

melhorando

Quais os problemas da solução anterior?

index.js - continuando

Apague o esse trecho:

```
const lista_produtos = require('./dados.json’)
```

E coloque esse:

```
const fs = require('fs')
const arquivo = fs.readFileSync('./dados.json', 'utf8')
const lista_produtos = JSON.parse(arquivo)
```

Banco de dados

Dados de um array

```
const lista_produtos = [
    {
        "titulo": "Red Nike",
        "foto":"foto.jpg",
        "descricao": "",
        "preco": 399.00,
        "avaliacao": 4
    }
]
```

Dados de um json

```
const lista_produtos = require('./dados.json')
```

Dados do banco de dados - conexão

```
// usando a biblioteca mysql
let conexao = mysql.createConnection({
  host: “endereço do banco de dados”,
  user: “usuário do banco de dados”,
  password: “senha do usuário”,
  database: “nome do banco de dados”
});
```

Dados do banco de dados – testando a conexão

```
conexao.connect(function (erro) {})
```

Dados do banco de dados – testando a conexão

```
conexao.connect(function (erro) {
  if (erro) {
    console.log("Deu ruim na conexão \n");
    throw erro;
  } else {
    console.log("Conexão deu bom \n")
  }
})
```

Dados do banco de dados – consultando dados

```
conexao.query(“comandos sql”, function (erro, dados, campos) {})
```

Dados do banco de dados – consultando dados

```
conexao.query("SELECT * FROM produtos", function (erro, dados, campos) {
    console.log(dados)
    res.send(dados)
  })
```

No terminal

```
npm install mysql
```

index.js - conexão

Apague o esse trecho:

```
const lista_produtos = require('./dados.json’)
```

E coloque esse:

```
let mysql = require('mysql')
let conexao = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "",
  database: "bd_loja"
})
```

index.js – conexão - continuando

```
conexao.connect(function (erro) {
  if (erro) {
    console.log("Deu ruim na conexão \n");
    throw erro;
  } else {
    console.log("Conexão deu bom \n")
  }
})
```

index.js – selecionando dados

Apague o esse trecho:

```
res.send(lista_produtos)
```

E coloque esse:

```
conexao.query("SELECT * FROM produtos", function (erro, lista_produtos, campos) {
    console.log(lista_produtos);
    res.send(lista_produtos)
  })
```

atividades

objetivo: criar uma para exibir em cards(com nome da unidade, foto, endereço, email, telefone) todas as 6 unidades

1 - duplicar a index.html e renomear para unidades.html

2 - duplicar o app.js e renomear para app-unidades.js

3 - no index.js criar a rota /unidades

4 - na rota /unidades fazer o select para retornar as unidades

5 - fazer os ajustes em unides.html, app-unidades.js e index.js necessários para funcionar a o página unidades.html

Teste o seu site para ver se os produtos estão sendo listados

Teste o seu site para ver se os produtos estão sendo listados

Node.js

Páginas de categorias
Pegando dados da url
Recebendo parâmetros
Rotas dinâmicas

FRONTEND

Vocês fizeram

```
<a class="nav-link" href=“unidades.html">Unidades</a>
```

```
fetch('http://localhost:3000/unidades/', { method: 'GET’ })
```

Eu poderia fazer

```
<a class="nav-link" href=“tenis.html">Tênis</a>
<a class="nav-link" href=“camisetas.html">Camisetas</a>
<a class="nav-link" href=“oculus.html">Óculos</a>
```

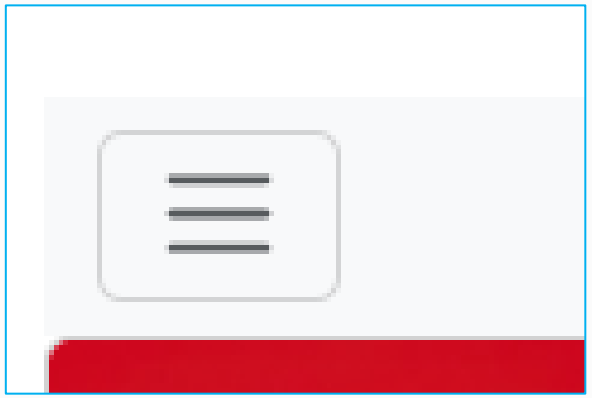
```
fetch('http://localhost:3000/tenis/', { method: 'GET’ })
fetch('http://localhost:3000/camisetas/', { method: 'GET' })
fetch('http://localhost:3000/oculus/', { method: 'GET' })
```

Adicionando menu - FrontEnd

Renomeie o arquivo **index.html** para **produtos.html**

Entre no site do Bootstrap(5) e pesquise o componente **navbar**

Crie a barra de navegação igual à essa:



Não esqueça que pra navebar funcionar a responsividade **precisa adicionar o Javascript do bootstrap no final da página**.

Link dos menus

```
<a class="nav-link" href="produtos.html">Todos os produtos</a>
<a class="nav-link" href="produtos.html?categoria=tenis">Tênis</a>
<a class="nav-link" href="produtos.html?categoria=camisetas">Camisetas</a>
<a class="nav-link" href="produtos.html?categoria=oculos">Óculos</a>
<a class="nav-link" href="unidades.html">Unidades</a>
```

Como eu vou conseguir pegar a indicação da categoria (`categoria=camisetas`) no frontend e enviar para o backend?

Pegar partes da url

```
window.location

window.location.search

const parametros = new URLSearchParams(window.location.search)

const palavra = parametros.get('keywords')
```

app.js

Nome começo da função fnCarregarDados() adicione o código:

```
function fnCarregarDados() {
    const parametros = new URLSearchParams(window.location.search)
     const existe_categoria = parametros.has('categoria')

     let rota_categoria = ""
     if (existe_categoria) {
         rota_categoria = parametros.get('categoria') + "/"
}
    fetch('http://localhost:3000/produtos/' + rota_categoria, { method:
'GET' })
```

BACKEND

Vocês fizeram

app.get("/unidades/", function (req, res) { })

Eu poderia fazer

```
app.get("/camisetas/", function (req, res) { })

app.get("/tenis/", function (req, res) { })

app.get("/oculus/", function (req, res) { })
```

Pegar informações de parâmetros GET

```
app.get("/item/:id", function (req, res) {
    const id = req.params.id
    res.send(id)
})
```

index.js

Adicione o Código abaixo após a rota `/produtos` :

```
// Read by categoria - [GET] /produtos/:categoria
app.get("/produtos/:categoria", function (req, res) {
    res.setHeader('Access-Control-Allow-Origin', '*')
    // pegamos a categoria que foi enviada na requisição
    const categoria = req.params.categoria
    conexao.query(`SELECT * FROM produtos where categoria='${categoria}'`, function (erro,
    dados, campos) {
        res.send(dados)
    })
})
```

atividades

No html criar links. Ex:

```
<a href=“produtos.html?categoria=camisetas&ordem=preco”>Ordenar pelo preço</a>
<a href=“produtos.html?categoria=camisetas&ordem=titulo”>Ordenar pelo titulo</a>
```

Criar uma nova rota:

```
/produtos/:categoria/:ordem
```

Aonde você deve passar a ordem esperada da listagem. Exemplo:

```
/produtos/camisetas/preco – a listagem deve ser feita na ordem do preço menor para maior
/produtos/camisetas/titulo – a listagem deve ser feita na ordem alfabética pelo título
```

Node.js

CORS
Cadastrar produto

Você lembra disso?

```
res.setHeader('Access-Control-Allow-Origin', '*')
```

O que é?

É utilizado em servidores web para permitir que qualquer domínio (origem) acesse os recursos (como dados de uma API) fornecidos por esse servidor, superando a política de segurança padrão dos navegadores conhecida como **CORS (Cross-Origin Resource Sharing)**

Aonde estamos usando?

Porque estamos usando?

Vamos mudar

Apague toda a linha com esse código:

```
res.setHeader('Access-Control-Allow-Origin', '*')
```

No terminal:

npm install cors

No começo do nosso código, abaixo da linha `const app = express()` coloque:

```
// necessário para permitir requisições de diferentes origens(domínios/servidores)
const cors = require('cors')
app.use(cors())
```

CORS - Cross-Origin Resource Sharing

É um mecanismo de segurança utilizado pelos navegadores para permitir que recursos (APIs, fontes, imagens) de um domínio (origem) sejam acessados por scripts rodando em outro domínio. Ele contorna a "Política de Mesma Origem" (Same-Origin Policy), autorizando requisições legítimas entre origens diferentes.

Erro Comum: Ocorre quando um front-end (localhost:5500) tenta acessar uma API (localhost:3000) sem que o servidor desta API tenha configurado os cabeçalhos CORS adequados.

Sem o CORS, navegadores bloqueiam solicitações fetch para domínios diferentes, restringindo a integração de aplicações modernas.

Teste o seu site para ver nada quebrou

Cadastro de produtos

FrontEnd

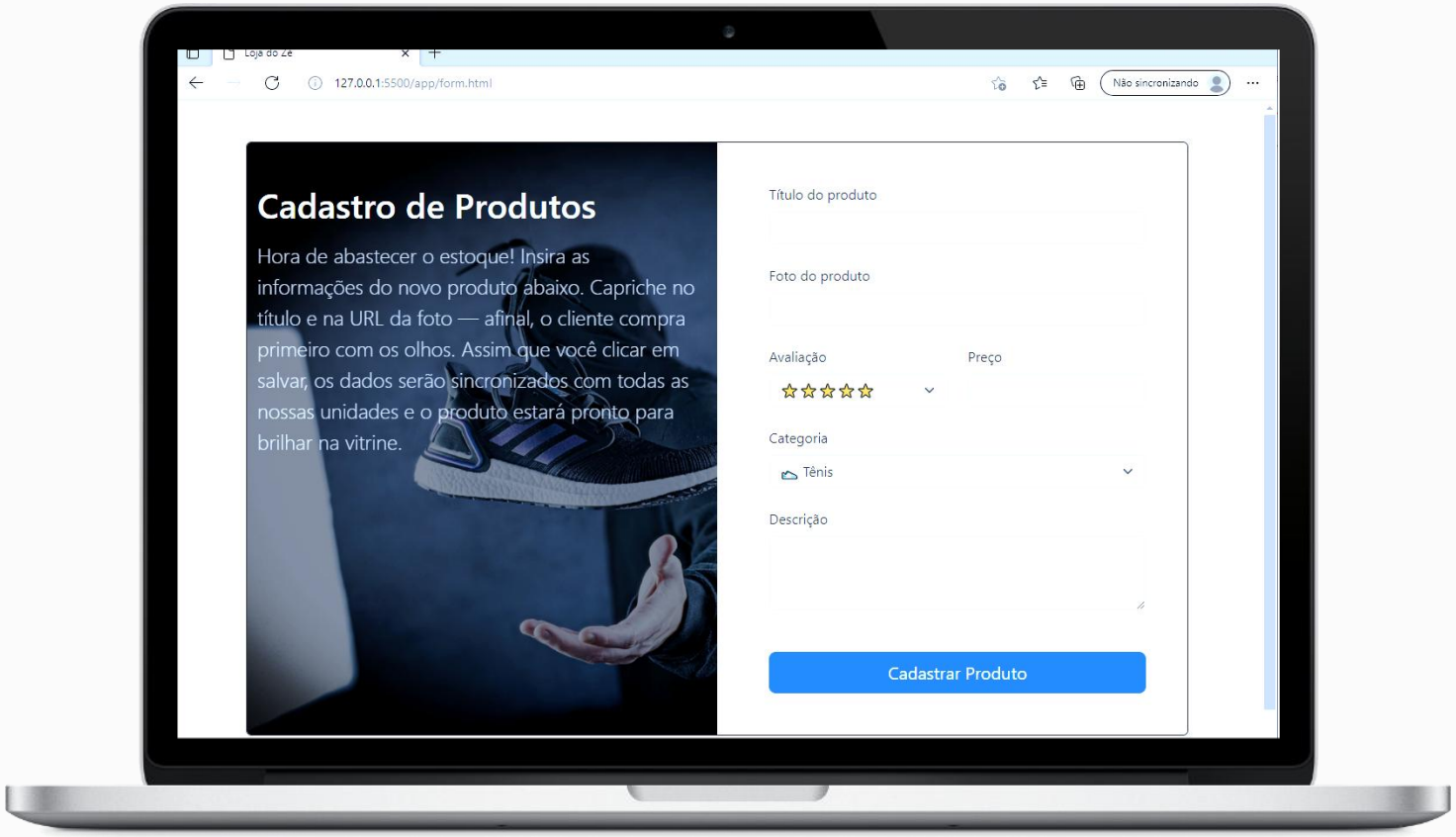
- Criar o formulário
- Enviar os dados do formulário para a API

BackEnd

- rota /produto[POST]
  - criar a rota
  - receber os dados do frontend
  - montar o sql de insert
  - executar o insert e retornar resposta para o frontend

FRONTEND

Formulário de cadastro



FrontEnd

Dentro da pasta `site-loja` crie os arquivos:

- `form.html`
- `app-form.js`

Cole o código fornecido em cada um dos arquivos:

app-form.js - fnCadastrarProdutos()

```
let formDados = {
        titulo: document.getElementById("_____").value,
        preco: document.getElementById("_____").value,
        descricao: document.getElementById("_____").value,
        avaliacao: document.getElementById("_____").value,
        foto: document.getElementById("_____").value,
        categoria: document.getElementById("_____").value
    }
```

app-form.js - fnCadastrarProdutos()

```
fetch('http://localhost:3000/produto/', {
          method: 'POST’,
          headers: { 'Content-Type': 'application/json’ },
          body: JSON.stringify(formDados)
        })
  .then(reposta => reposta.json())
  .then((dados) => {
    fnLimparCampos()
    console.log(dados)
   })
   .catch(erro => console.log(erro.message))
}
```

funções

```
function mensagem(){ alert(‘oi’) } // função

function(){alert(‘oi’)} // função anônima

() => {alert(‘oi’)} // arrow function
```

funções

```
btn_salvar.addEventListener("click", function salvar() {
    fnCadastrarProdutos()
})

btn_salvar.addEventListener("click", function() {
    fnCadastrarProdutos()
})

btn_salvar.addEventListener("click", () => {
    fnCadastrarProdutos()
})
```

BACKEND

index.js

Vamos criar o verbo app.post("/produto/"

index.js

```
app.post("/produto/", function (req, res) {
  const { titulo, preco, descricao, avaliacao, foto, categoria } = req.body;
  conexao.query(`
    INSERT INTO produtos(titulo, foto, descricao, preco, avaliacao, categoria)
    values('${titulo}','${foto}','${descricao}',${preco}, ${avaliacao}, '${categoria}')`,
    function (erro, resultado) {
      if (erro) {
        res.json(erro);
      }
      res.send(resultado.insertId);
  });
})
```

atividades

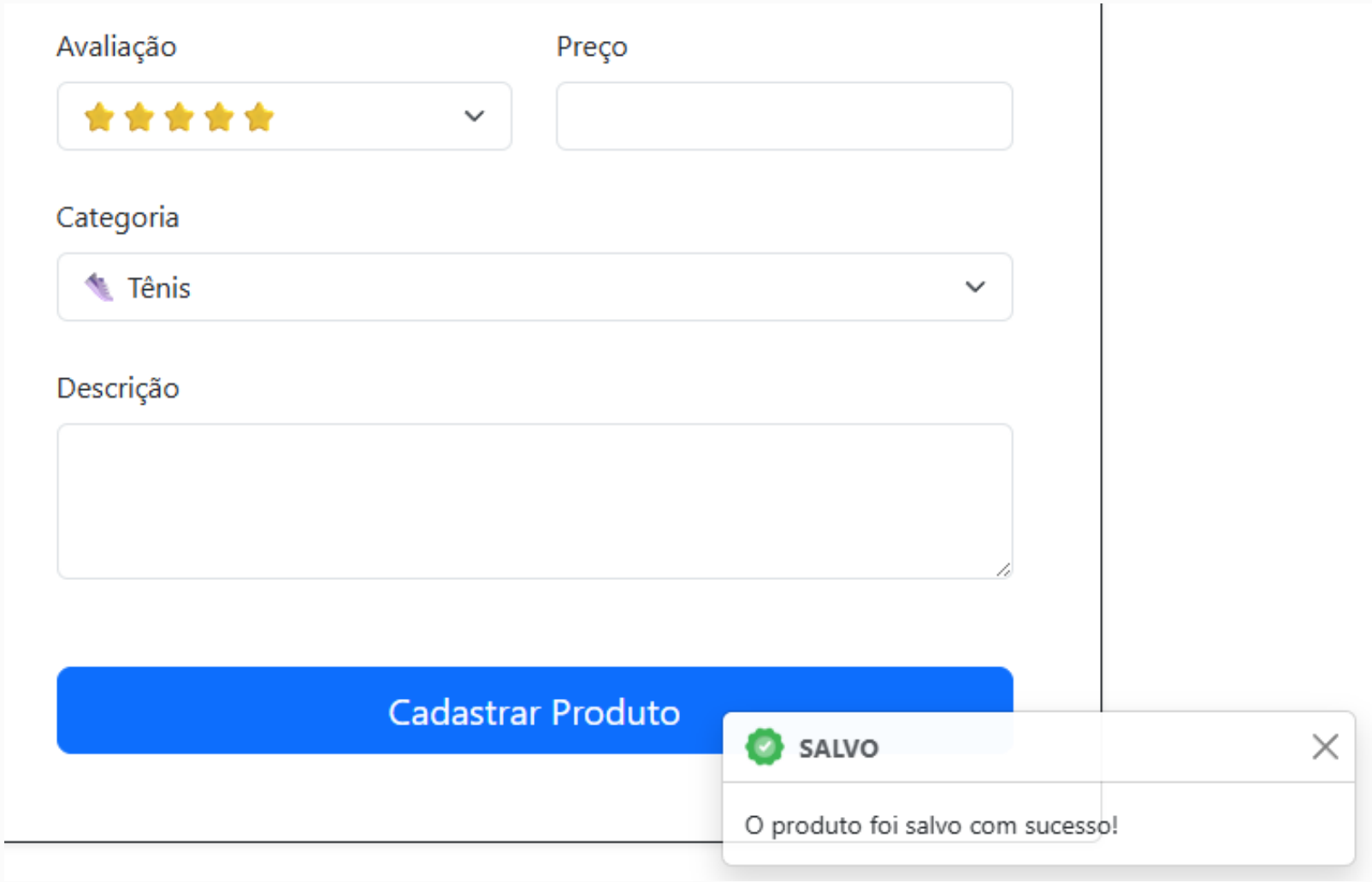
Crie a tela de cadastro de unidades

Node.js

Notificações

Senac

Toast (notificação)



form.html

Acrescente o código abaixo depois da última ** :**

```
<div class="toast">
  <div class="toast-header">
    SUCESSO
    <button type="button" class="btn-close" data-bs-dismiss="toast"></button>
  </div>
  <div class="toast-body">
    O produto foi salvo com sucesso!
  </div>
</div>
```

app-form.js

Acrescente o código abaixo antes da função **fnCadastrarProdutos():**

```
function fnMensagemSalvar() {
    var toastElList = [].slice.call(document.querySelectorAll('.toast'))
    var toastList = toastElList.map(function (toastEl) {
        return new bootstrap.Toast(toastEl)
    })
    toastList.forEach(toast => toast.show())
}
```

app-form.js

Acrescente o código abaixo dentro da função **`fnCadastrarProdutos()`** depois da linha **`fnLimparCampos();`**

```
fnMensagemSalvar()
```

Node.js

Refatoração de código
Body-parser

index.js

```
app.post("/produto/", function (req, res) {
  const { titulo, preco, descricao, avaliacao, foto, categoria } = req.body;
  conexao.query(`
    INSERT INTO produtos(titulo, foto, descricao, preco, avaliacao, categoria)
    values('${titulo}','${foto}','${descricao}',${preco}, ${avaliacao}, '${categoria}')`,
    function (erro, resultado) {
      if (erro) {
        res.json(erro);
      }
      res.send(resultado.insertId);
  });
})
```

index.js

```
app.post("/produto/", function (req, res) {
  const data = req.body
  conexao.query('INSERT INTO produtos set ?', [data], function (erro, resultado) {
    if (erro) {
      res.send(erro)
    }
    res.send(resultado.insertId)
  });
})
```

ENCODED

```
No terminal

# npm install body-parser
```

No index.js, antes da linha `const cors = require('cors')` adicione as linhas:

```
const bodyParser = require('body-parser')

app.use(bodyParser.urlencoded({ extended: true }))

app.use(bodyParser.json())
```

Node.js

Login

BACKEND

index.js

Vamos criar a rota `app.post("/login/"`

```
app.post("/login/", function (req, res) {
  const usuario = req.body.usuario
  const senha = req.body.senha
  conexao.query(`select * from usuarios where usuario = '${usuario}' and senha =
'${senha}'`, function (erro, resultado, campos) {
    if (erro) {
      res.send(erro)
    }else{
      if (resultado.length > 0) {
        res.sendStatus(200)
      } else {
        res.sendStatus(401)
      }
    }
  })
})
```

Thunder Client

Abra o Thunder Client e crie um requisição com as configurações abaixo:

Método: POST

url: http://localhost:3000/login

Body Type: JSON

Body Content:

```
{
    "usuario" : "admin@admin.com.br",
    "senha": "123"
}
```

FRONTEND

FrontEnd

Na pasta do FrontEnd crie os arquivos:

- `login.html`
- `app-login.js`

Copie os conteúdos para dentro dos arquivos.

fnFazerLogin()

```
function fnFazerLogin() {
    let formDados = {
        usuario: document.getElementById("_________").value,
        senha: document.getElementById("_________").value
    }
    fetch('http://localhost:3000/login/', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
        },
        body: JSON.stringify(formDados)
    })
        .then(resposta => resposta.status)
        .then((dados) => {
            fnLimparCampos()
            console.log(dados)
        })
        .catch(erro => console.log(erro.message))
}
```

Node.js

Cadastro de usuário

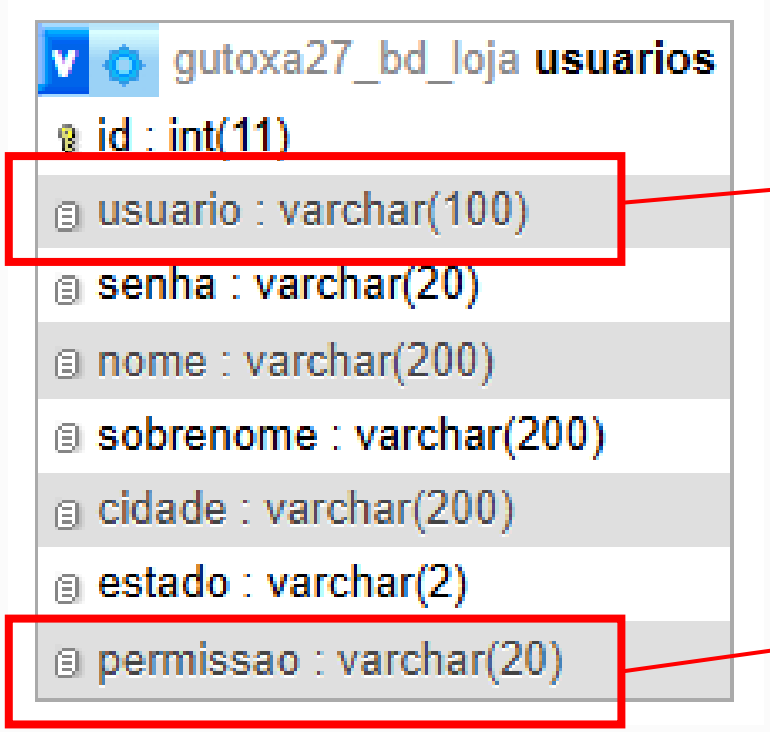
Senac

FRONTEND

FrontEnd e BackEnd

FRONTEND - Crie o formulário necessário para registrar um novo usuário

BACKEND – Crie a rota para cadastrar o usuário



O usuário é o email:
guto@guto.com

Permissão pode ser apenas 2 valores:
admin
vendedor

HTTP STATUS

httpstatus.com.br

200 OK: A requisição foi bem-sucedida.

401 Unauthorized: Autenticação necessária ou falha.

500 Internal Server Error: Erro genérico do servidor.

Node.js

admin

FRONTEND

FrontEnd

Na pasta do site-loja crie os arquivos:

- `admin.html`
- `app-admin.js`
- `style-admin.css`

Copie os conteúdos para dentro dos arquivos.

app-admin.js - fnCarregarDados

```
function fnCarregarDados() {
    fetch('http://localhost:3000/produtos/' , { method: 'GET' })
        .then((resposta) => resposta.json())
        .then((produtos) => {
            produtos.forEach(produto => {
                fnMontarLinhaProduto(produto)
            });
        })
        .catch(err => console.log(err.message))
}
```

app-admin.js - fnMontarLinhaProduto

```
function fnMontarLinhaProduto(produto) {
    let cartao = `
        <tr>
            <td>
                <img src="${produto.foto}" width="50" alt="${produto.nome}">
            </td>
            <td>${produto.id}</td>
            <td>${produto.titulo}</td>
            <td>${produto.descricao}</td>
            <td>${produto.categoria}</td>
            <td>R$ ${produto.preco}</td>
            <td>${produto.avaliacao}</td>
```

app-admin.js - fnMontarLinhaProduto

```
</tr>
    `

    document.querySelector("#lista-produtos").innerHTML += linha
}


fnCarregarDados()
```

app-admin.js – melhorando seu código

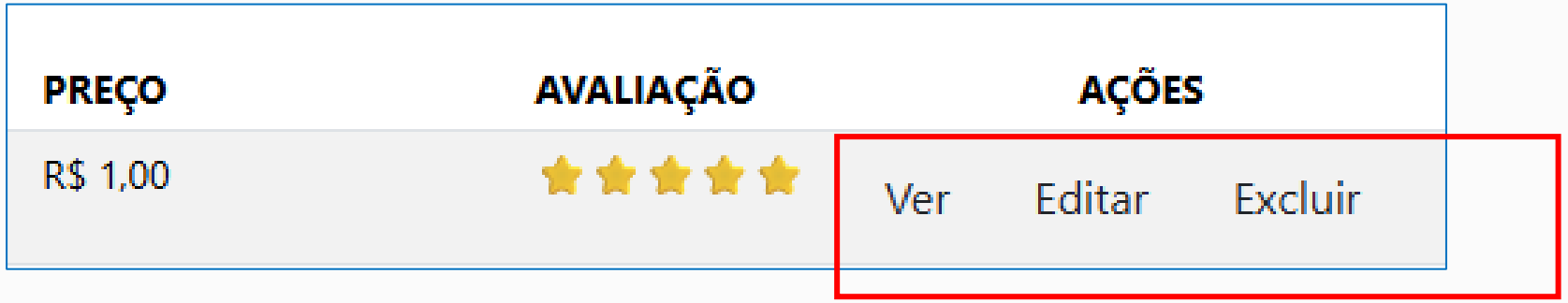
```
<td>${produto.titulo.substring(0, 20)}</td>
<td>${produto.descricao.substring(0,50)}</td>
<td>${produto.preco.toLocaleString('pt-BR', { style: 'currency', currency: 'BRL'})}</td>
<td>${"⭐".repeat(produto.avaliacao)}</td>
```

FrontEnd

Criar a coluna de título **ações**

PREÇO	AVALIAÇÃO	AÇÕES

FrontEnd



Criar dois links e um botão

app-admin.js – fnMontarLinhaProduto

```
<td>
    <a href="um-produto.html?id=${produto.id}" class="btn" >Ver</a>
    <a href="editar-produto.html?id=${produto.id}" class="btn" >Editar</a>
    <button type="button" class="btn">Excluir</a>
</td>
```

index.js

```
// Create - [POST] /produto
app.post("/produto/", function (req, res) {})

// Read All - [GET] /produtos
app.get("/produtos", function (req, res) {})

// Read ONE - [GET] /produto/:id
app.get("/produto/:id", function (req, res) {})

// Update - [PUT] /produto/:id
app.put("/produto/:id", function (req, res) {})

// Delete - [DELETE] /produto/:id
app.delete("/produto/:id", function (req, res) {})
```

Node.js

admin – ver um produto

BACKEND

index.js

```
// Read ONE - [GET] /produto
app.get("/produto/:id", function (req, res) {
  const id = req.params.id
  conexao.query("SELECT * FROM produtos where id = ? ", [id] ,
function (erro, dados, campos) {
    res.json(dados)
  })
})
```

DEPOIS QUE FIZER A ROTA, TESTE COM THUNDERCLIENT

FRONTEND

FrontEnd

Na pasta do site-loja crie os arquivos:

- `um-produto.html`
- `app-um-produto.js`

Copie os conteúdos para dentro dos arquivos.

app-um-produto.js - fnCarregarDados

```
function fnCarregarDados() {
    const parametros = new URLSearchParams(window.location.search);
    const id = parametros.get('id') + "/"
    fetch('http://localhost:3000/produto/' + id, { method: 'GET' })
        .then((resposta) => resposta.json())
        .then((produtos) => {
            produtos.forEach(produto => {
                fnMontarProduto(produto)
            })
        })
        .catch(err => console.log(err.message))
}
```

app-um-produto.js - fnMontarProduto

```
function fnMontarProduto(produto) {
    document.getElementById("foto").src = produto.foto
    document.getElementById("titulo").innerHTML = produto.titulo
    document.getElementById("descricao").innerHTML = produto.descricao
    document.getElementById("categoria").innerHTML = produto.categoria
    document.getElementById("preco").innerHTML = produto.preco.toLocaleString('pt-BR', {
style: 'currency', currency: 'BRL'})
    document.getElementById("avaliacao").innerHTML = "⭐".repeat(produto.avaliacao) +
`(${produto.avaliacao})`
}

fnCarregarDados()
```

Node.js

admin – alterar um produto

FRONTEND

Pasta site-loja

Vamos utilizar a estrutura do cadastro de produto pois o cadastro é:

`FORM EM BRANCO + FUNÇÃO DE CADASTRO`

A atualização é:

`FORM PREENCHIDO + FUNÇÃO DE ATUALIZAR`

Copie o arquivo **form.html** e altere o nome para **editar-produto.html**

Copie o arquivo **app-form.js** e altere o nome para **app-produto-editar.js**

No arquivo **editar-produto.html** faça referência ao arquivo **app-produto-editar.js**

app-produto-editar.js – criar função fnCarregarDados()

```
function fnCarregarDados() {
    const parametros = new URLSearchParams(window.location.search);
    const id = parametros.get('id') + "/"

    fetch('http://localhost:3000/produto/' + id, { method: 'GET' })
        .then((resposta) => resposta.json())
        .then((produtos) => {
            produtos.forEach(produto => {
                fnMontarProduto(produto)
            })
        })
        .catch(err => console.log(err.message))
}
```

app-produto-editar.js – criar função fnMontarProduto

```
function fnMontarProduto(produto) {
    document.getElementById("fundo-imagem").style.backgroundImage = `url(${produto.foto})`
    document.getElementById("foto").value = produto.foto
    document.getElementById("titulo").value = produto.titulo
    document.getElementById("descricao").value = produto.descricao
    document.getElementById("categoria").value = produto.categoria
    document.getElementById("preco").value = produto.preco
    document.getElementById("avaliacao").value = produto.avaliacao
}

fnCarregarDados()
```

app-produto-editar.js

Altere o nome da função `fnCadastrarProdutos` para `fnSalvarProdutos`

app-produto-editar.js

```
function fnSalvarProdutos() {
    const parametros = new URLSearchParams(window.location.search);
    const id = parametros.get('id') + "/"

    let formDados = {
        titulo: document.getElementById("titulo").value,
        preco: document.getElementById("preco").value,
        descricao: document.getElementById("descricao").value,
        avaliacao: document.getElementById("avaliacao").value,
        foto: document.getElementById("foto").value,
        categoria: document.getElementById("categoria").value
    }
```

app-produto-editar.js

```
    fetch('http://localhost:3000/produto/' + id, {
        method: 'PUT',
        headers: {
            'Content-Type': 'application/json',
        },
        body: JSON.stringify(formDados)
    })
        .then(reposta => reposta.json())
        .then((dados) => {
            fnMensagemSalvar()
        })
        .catch(erro => console.log(erro.message))
}
```

BACKEND

index.js

```
// Update - [PUT] /produto/:id
app.put("/produto/:id", function (req, res) {
  const id = req.params.id
  const data = req.body

  conexao.query(`UPDATE produtos set ? where id = ${id}`, [data], function (erro, resultado)
{
    if (erro) {
      res.send(erro)
    }
    res.send({"status":200, "message": "Atualizado com sucesso!"})
  })
})
```

Node.js

admin – excluir um produto

BACKEND

index.js

```
// Delete - [DELETE] /produto/:id
app.delete("/produto/:id", function (req, res) {
  const id = req.params.id
  conexao.query(`delete from produtos where id = ${id}`, function (erro, resultado) {
    if (erro) {
      res.send(erro)
    }
    res.json({ "status": 200, "message": "Excluído com sucesso!" })
  })
})
```

FRONTEND

Pasta site-loja

Crie o arquivo **app-produto-excluir.js**

app-produto-excluir.js

```
function fnExcluirProduto(id) {
    fetch('http://localhost:3000/produto/' + id, {
            method: 'DELETE',
            headers: {
                'Content-Type': 'application/json',
            }
        })
            .then(reposta => reposta.json())
            .then((dados) => {
                console.dir(dados)
            })
            .catch(erro => console.log(erro.message))
}
```

app-admin.js – fnMontarLinhaProduto()

```
<button type="button" class="btn" onclick="fnExcluirProduto(${produto.id})">Excluir</button>
```

admin.html

```
<script src="app-produto-excluir.js"></script>
```

Node.js

admin – confirmar exclusão

FRONTEND

app-produto-excluir.js

```
function fnExcluirProduto(id) {
    let confirmar = confirm("Podemos realmente excluir?")
    if (confirmar) {
        fetch('http://localhost:3000/produto/' + id, {
        method: 'DELETE',
        headers: {
            'Content-Type': 'application/json',
        }
      })
        .then(reposta => reposta.json())
        .then((dados) => {
            console.log(dados)
        })
        .catch(erro => console.log(erro.message))
    }
}
```

Node.js

admin – removendo a linha

FRONTEND

app-admin.js

```
<button type="button" class="btn" onclick="fnExcluirProduto(
${produto.id} , event.target)">Excluir</button>
```

app-produto-excluir.js

```
function fnExcluirProduto(id, elemento) {
      let confirmar = confirm("Podemos realmente excluir?")
    if (confirmar) {
        fetch('http://localhost:3000/produto/' + id, {
            method: 'DELETE',
            headers: {
                'Content-Type': 'application/json',
            }
        })
            .then(reposta => reposta.json())
                .then((dados) => {
                    elemento.closest("tr").remove()
                })
                .catch(erro => console.log(erro.message))
    }
}
```