CAPÍTULO 3 – Serviço HTTP e renderização de HTML

Neste capítulo você vai ver:

- Como criar um servidor HTTP utilizando o módulo nativo “http”;
- Como capturar parâmetros na requisição utilizando o módulo “url”;
- Como ler e gravar arquivos com o modulo nativo “fs”; e
- Como renderizar arquivos HTML combinando os mósulos “http”, “url” e ”fs” .

Atividade 1 – Conhecer o módulo “fs”

Essa atividade tem como objetivo:

- Ler e gravar arquivos utilizando o core module “fs” (file system) que tem como função acessar os recursos do sistema de arquivos;
- Recapitular as funções de manipulação de dados no formato JSON.

Comandos: fs.readFileSync() | fs.writeFileSync() | JSON.parse() | JSON.stringify()

Ler inserir e gravar em arquivos JSON.

Vamos aprender a ler arquivos gravar arquivos utilizando o módulo “fs”. Vamos também aproveitar e revisar a sintaxe e as funções de manipulação de dados no formato JSON.

1. Para iniciar as atividades do capítulo 03, crie um diretório na pasta “NODEJS_EXERCÍCIOS” com o nome “Cap03”
2. Para este exercício iremos utilizar um arquivo JSON para ser lido e gravado e dois módulos (script principal e modulo com funções exportadas). Por isso, crie um subdiretório em “Cap03” chamado “**at01-fsLerGravarJSON**”.
3. Dentro do subdiretório “**at01-fsLerGravarJSON**” crie um arquivo chamado “**alunos.json**” e insira o seguinte código utilizando a sintaxe JSON:

```
[
   {
     "nome": "Carlos",
     "nota1": 5.5,
     "nota2": 6,
     "nota3": 7,
     "nota4": 8
   }
]
```

4. Agora vamos criar o módulo que abrigará as funções que implementam as necessidades do programa. Para isso crie outro arquivo chamado “**fsJSON.js**” e insira o seguinte código:

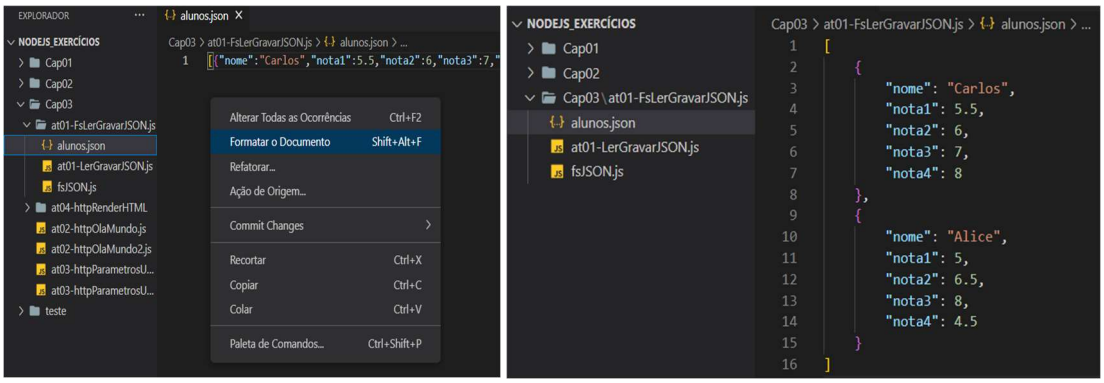
```
const fs = require('fs')
module.exports = {
  lerJSON: (arquivo)=>{
     //Lê arquivos de texto de forma síncrona
     let json = fs.readFileSync(arquivo, 'utf8')
     return json
  },
  converterJSON_Obj: (json)=>{
     //Converte string JSON para um objeto javaScript
     let dados = JSON.parse(json)
     return dados
  },
  converterObj_JSON:(dados)=>{
     //Converte objeto JavaScript para uma string JSON
     let json = JSON.stringify(dados)
     return json
  },
  salvarJSON:(json, arquivo)=>{
     //Grava arquivos de texto de forma síncrona
     fs.writeFileSync(arquivo, json)
  }
}
```

5. Agora crie um módulo chamado "**at01-LerGravarJSON.js**" e insira o código a seguir:

```
const rwJSON = require('./fsJSON') //Importação do módulo criado fsJSON
let arquivo = './alunos.json' //Dados a serem gravados
      let obj = {
            nome: 'Alice',
            nota1:5,
            nota2:6.5,
            nota3:8,
            nota4:4.5
      }
var json = rwJSON.lerJSON(arquivo) //Leitura e carga de um arquivo JSON
console.log(json)
//Conversão de string JSON em um objeto javaScript
      var dados = rwJSON.converterJSON_Obj(json)
      console.log(dados)
//Inserção de novo registro
      dados.push(obj)
      console.log(obj)
//Conversão do obj atualizado para JSON
      json = rwJSON.converterObj_JSON(dados)
      console.log(json)
//Regravação do arquivo com dados atualizados
   rwJSON.salvarJSON(json, arquivo)
```

6. Execute o script do módulo agora usando o comando "**nodemon at01-LerGravarJSON.js**" e verifique no arquivo "**alunos.json**" que um novo registro foi adicionado, clique com o botão

direito no arquivo e escolha a opção Formatar documento ou o atalho Shift+Alt+F, conforme observado abaixo:



Atividade 2 – Implementação de um servidor HTTP simples

Essa atividade tem como objetivo:

- Se familiarizar com o core module “http”;

- Compreender funcionamento de requisições e respostas HTTP.

Comandos: http.createServer() | res.setHeader() | res.end() |server.listen()

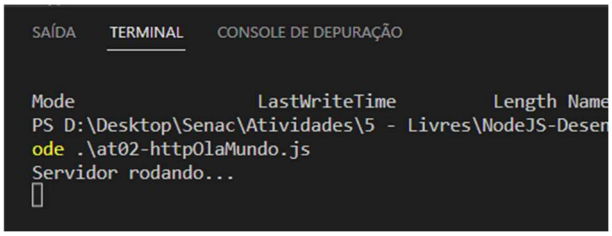
. “Olá mundo!” no navegador, utilizando um serviço “http”

Vamos aprender a ler arquivos gravar arquivos utilizando o módulo “fs”. Vamos também aproveitar e revisar a sintaxe e as funções de manipulação de dados no formato JSON.

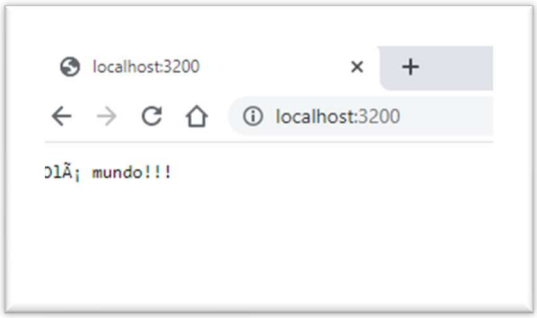
1. No subdiretório “Cap03”, crie um arquivo **“at02-OlaMundo.js”** e insira o seguinte código:

```
const http = require('http')

 //Cria o serviço HTTP, e passa callback que processa as requisições.
   const server = http.createServer((req, res)=>{

   //Configura o cabeçalho da resposta
         res.setHeader('Contenty-Type', 'text/html')
   //Responde a requisição
         res.end(`Olá mundo!!!`)
   })
 //Ativa o servidor para escutar as requisições
   server.listen('3200', ()=>{console.log('Servidor rodando...')})
```

2. Execute o script e observe que ao executar o script não é finalizado, ao invés disso, ele permanece em execução e mantem o serviço ativo:



3. Para realizar uma requisição, abra o navegador e **acesse a url "localhost:3200"** e perceba a resposta do servidor:

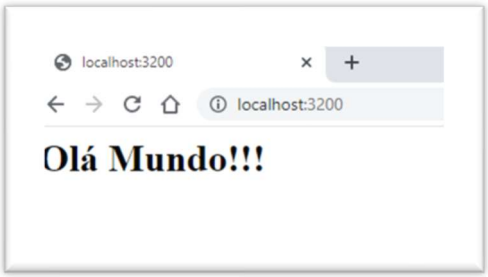


4. Volte o código e altere a frase de "Olá mundo!!!" para "Olá Node.js!!!". Atualize a página no navegador e perceba que a resposta continua sendo "Olá Mundo!!!";
5. Isto acontece devido a necessidade de reiniciar o serviço para que as alterações sejam aplicadas. Para isso, vá para o **terminal do VS Code** e pressione o **atalho Ctrl+C** para encerrar a execução do serviço.
6. Execute o código novamente e perceba que agora a resposta foi atualizada.
7. Vamos agora aperfeiçoar o código respondendo agora um código HTML. Substitua a string no método na chamada da função res.end() conforme código HTML abaixo, note que para:

```
  const http = require('http')

//Cria o serviço HTTP, com callback que processa e responde as requisições.
  const server = http.createServer((req, res)=>{
    //Configura o cabeçalho da resposta
      res.setHeader('Contenty-Type', 'text/html')
    //Responde a requisição
      res.end(`
        <head> <meta charset="UTF-8"></head>
        <body>
            <h1>Olá Mundo!!!</h1>
        </body>
      `)
  })
//Ativa o servidor para escutar as requisições
  server.listen('3200', ()=>{console.log('Servidor rodando...')})
```

Obs.: Para a manter a formatação de quebra de linhas e tabulação na string de resposta deve-se utilizar `crase` e não 'aspas'.

8. Pare o servidor e execute novamente o script. Acesse novamente o localhost:3200 no navegador e observe o resultado:



Atividade 3 – Passagem de parâmetros na url da requisição

Essa atividade tem como objetivo:

- Passar parâmetros para o servidor HTTP por meio da url da requisição;

Comandos: url.parse()

Passagem de parâmetros na url da requisição

Passar e ler parâmetros para o servidor utilizando a url da requisição utilizando o módulo "url".

1. No subdiretório "Cap03", crie um arquivo **"at03-httpParametrosUrl.js"** e insira o código:

```
const http = require('http')
const url = require('url')
const server = http.createServer((req, res)=>{
 //Converte parâmetros passados na url
      let parametro = url.parse(req.url, true)
      let nome = parametro.query.nome
      res.statusCode=200
      res.setHeader('Contenty-Type', 'text/html')
      if (nome){
            res.end(`<head> <meta charset="UTF-8"></head>
                    <body>
                        <h1>Olá ${nome}!!!</h1>
                    </body>`)
      } else {
      res.end(`<head> <meta charset="UTF-8"></head>
                <body>
                   <h1>Olá Node.js!!!</h1>
              <body>`)
    }
})
server.listen('3200', ()=>{console.log('Servidor rodando...')})
```

2. Realize testes executando o script e inicie o servidor. Abra o navegador e acesse com a seguinte url: “**localhost:3200**” e com “**localhost:3200/?nome=João**” e veja os resultados:

Atividade 4 – Renderizar arquivos HTML utilizando o módulo “fs”

Essa atividade tem como objetivo:

- Renderizar arquivos html utilizando o core module “fs”;

Comandos: pathname.substring() **|** fs.existsSync() **|** fs.readFile()

Renderizar arquivos HTML

Nesta atividade vamos fazer um serviço HTTP renderizar arquivos HTML. Para isso iremos utilizar um módulo javaScript e três arquivos HTML como exemplo, por isso iremos criar um subdiretório na pasta do capítulo 3.
Siga os passos a seguir:

1. No diretório “Cap03”, crie um subdiretório com o nome **“at04-HTTPFsRenderizarHTML”.**
2. Crie dentro do subdiretório os arquivos HTML que serão renderizados, conforme abaixo:

a) index.html

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
   <title>Index</title>
  </head>
  <body>
   <h1>Página Inicial</h1>
  </body>
</html>
```

b) usuario.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Usuário</title>
</head>
<body>
    <h1>A rota usuário foi acessada!</h1>
</body>
</html>
```

c) 404.html

```
<!DOCTYPE html>
<html lang="pt-br">
 <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Página não encontrada 404</title>
 </head>
 <body>
    <h1>Página não encontrada ERRO:404</h1>
 </body>
</html>
```

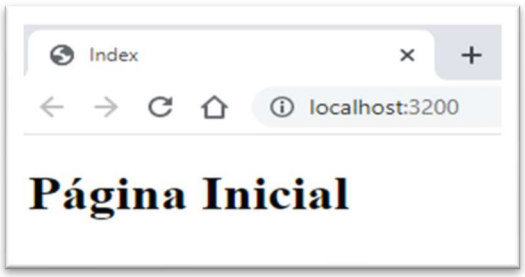
3. Agora faremos o script do serviço HTTP que receberá a requisição e renderizará as páginas.

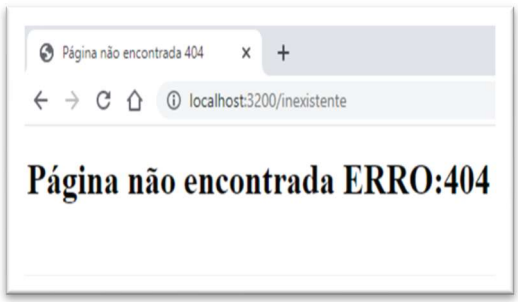
   Crie um arquivo com o nome "**at04-httpFsRenderizarHTML.js**" e insira o seguinte código:

```
const http = require('http')
const url = require('url')
const fs = require('fs')
const porta = 3200

const server = http.createServer((req, res)=>{
   const q= url.parse(req.url, true)
   //Obtem a substring após a "/"
   let pagina = q.pathname.substring(1)
   //Aponta url inicial para o index.html
   pagina = pagina==''? 'index.html':pagina
   //Insere a extensão HTML caso não for especificado
   pagina = !pagina.includes('html')? pagina=pagina+'.html': pagina
   console.log(página)

   if (fs.existsSync(pagina)){ //Verifica o arquivo existe
       fs.readFile(pagina, function(err, data){
```

```
            res.writeHead(200, {'Content-Type':'text/html'})
            res.write(data)
            return res.end()
        })
    } else {
        fs.readFile('404.html', function(err, data){
            res.writeHead(404, {'Content-Type':'text/html'})
            res.write(data)
            return res.end()
        })
    }
})
server.listen(porta, ()=>{
    console.log('Servidor rodando em: http://localhost:'+porta)
})
```

4. Execute o servidor e realize testes no navegador conforme imagens a seguir:





Exercícios extras

Estas atividades têm como objetivo aprofundar-se em conceitos importantes para a programação back-end.

Pesquisar e ler sobre:

1. Serviço HTTP e quais são as principais partes de uma requisição HTTP.
2. Diferença entre as estruturas de dados JSON e XML.
3. Sintaxe da estrutura de dados JSON.

Após as atividades de leitura, pode-se realizar:

4. Roda de conversa[1] para refletir e compartilhar conhecimento sobre a leitura.

[1] Ou outra ferramenta pedagógica proposta pelo docente.