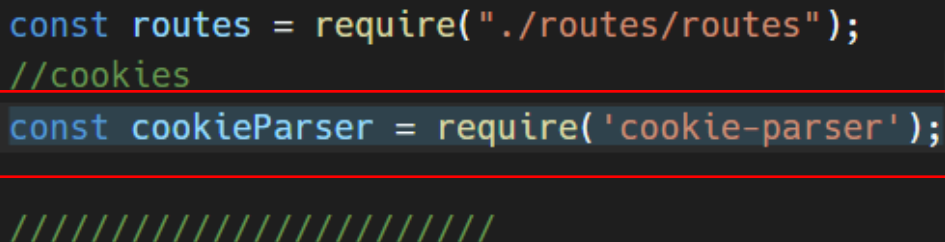


## Criando o carrinho de compras com cookies

Nesta aula vamos incluir no nosso projeto um carrinho que irá armazenar as informações em cookies e retornar nosso carrinho quando solicitado.

Vamos começar indo no nosso index.js da aplicação frontEnd e incluiremos a utilização de cookies.

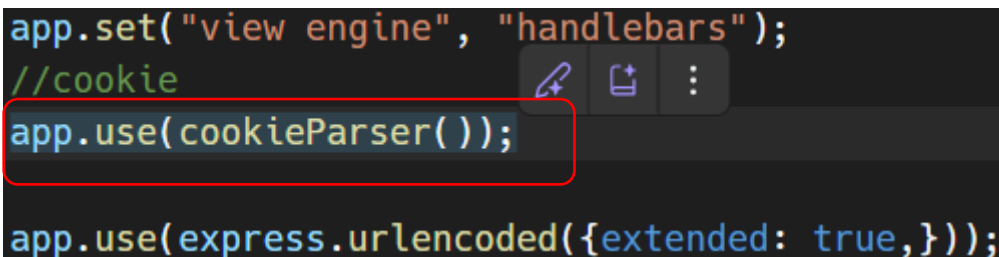
```
const cookieParser = require('cookie-parser');
```



```
const routes = require("../routes/routes");  
//cookies  
const cookieParser = require('cookie-parser');  
  
//////////
```

Ainda no arquivo index.js da aplicação FrontEnd, faremos a inclusão no app.use informando que iremos usar os cookies.

```
app.use(cookieParser());
```



```
app.set("view engine", "handlebars");  
//cookie  
app.use(cookieParser());  
  
app.use(express.urlencoded({extended: true}));
```

Pronto nosso arquivo index está pronto.

Vamos agora criar nossos serviços no services.js referente aos cookies.

### ADICIONAR UM COOKIE

Para adicionar um cookie, vamos criar uma função chamada Carrinho Adicionar.

Vamos ver o código linha alinhada.

```
static async CarrinhoAdicionar(req,res){  
  const Item = {  
    id: req.params.id,  
    nome: req.params.nome  
  };  
}
```

Na primeira parte estamos criando um objeto chamado item e guardaremos o id e o nome que serão enviados via rota por uma função.

Agora vamos verificar se já existe o cookie criado.

```
// Verificando se já existe um cookie para o carrinho  
if (req.cookies.carrinho) {  
  // Se já existe, adiciona o novo item  
  const carrinho = JSON.parse(req.cookies.carrinho);  
  carrinho.push(Item);  
  res.cookie('carrinho', JSON.stringify(carrinho), { maxAge: 900000, httpOnly:  
true });  
} else {  
  // Se não existe, cria um novo carrinho com o item  
  const carrinho = [Item];  
  res.cookie('carrinho', JSON.stringify(carrinho), { maxAge: 900000, httpOnly: true  
});  
}  
  
res.send('Item adicionado ao carrinho');  
  
}
```

Se já existir um cookie com o mesmo nome, usamos carrinho.push para incluir mais um item ao cookie. Senão existir o cookie, criamos e adicionamos os dados do carrinho.

## REMOVER ITEM DO COOKIE

```
// Rota para remover um item do carrinho  
static async CarrinhoRemoverItem(req,res){  
  const itemDeletar = req.params.item;  
  // Verificando se existe um cookie para o carrinho  
  if (req.cookies.carrinho) {  
    // Obtendo o carrinho atual do cookie  
    let carrinho = JSON.parse(req.cookies.carrinho);  
  }  
}
```

```
// Removendo o item do carrinho, se existir
carrinho = carrinho.filter(item => item.id !== itemDeletar);
// Atualizando o cookie com o carrinho modificado
res.cookie('carrinho', JSON.stringify(carrinho), { maxAge: 900000,
httpOnly: true });

res.send('Item removido do carrinho');
} else {
res.send('Carrinho vazio');
}
}
```

Ao chamar a função CarrinhoRemoverItem, faremos a verificação no item e removeremos somente o item do carrinho, deixando os demais no cookie.

## LISTAR COOKIE

A função CarrinhoListar exibe os dados do cookie e devolve em um objeto renderizado. Assim poderemos trabalhar nas views.

Ela é a mais simples, verificamos se o carrinho possui itens senão possuir, mostramos uma mensagem para o usuário informando que o carrinho está vazio.

```
static async CarrinhoListar(req,res){
// Rota para exibir o carrinho
if (req.cookies.carrinho) {
const carrinho = JSON.parse(req.cookies.carrinho);
res.render('carrinhos/Listar', {carrinho});
} else {
res.send('Carrinho vazio');
}
}
```

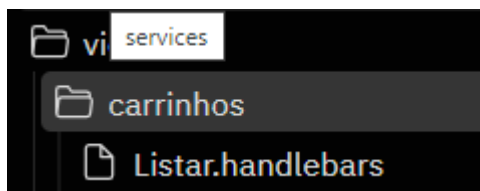
Vamos agora no arquivo routes e vamos criar nossas rotas.

Nossas rotas estão mostradas abaixo:

```
//rotas para os cookies  
router.get('/carrinho/Adicionar/:id/:nome',Services.CarrinhoAdicionar);  
router.get('/carrinho/listar', Services.CarrinhoListar);  
router.get('/carrinho/remover/:item', Services.CarrinhoRemoverItem);
```

```
//rotas para os cookies  
router.get('/carrinho/Adicionar/:id/:nome',Services.CarrinhoAdicionar);  
router.get('/carrinho/listar', Services.CarrinhoListar);  
router.get('/carrinho/remover/:item', Services.CarrinhoRemoverItem);  
  
module.exports=router;
```

Vamos agora criar uma pasta dentro da views chamada carrinhos e criar um arquivo chamado **Listar.handlebars**



Dentro do arquivo Listar.handlebars da pasta carrinhos vamos incluir o código abaixo:

```
<h1 class="text-center my-3">Lista de Produtos do Carrinho</h1>  
<center><table class="table">  
  <div class="row">  
    <div class="col-12">  
      <thead>  
      </thead>  
      <tbody>  
      <tr >
```

```

    {{#each carrinho}}
    <br>
    Código Produto:<strong> {{this.id}}</strong> - <strong>{{this.nome}}</strong>
    <a href="/carrinho/remover/{{this.id}}"></a>
    </th>
    <form class="borde-0 m-0" action="/carrinho/Adicionar" method="POST">
    <input type="hidden" name="id" value="{{this.id}}">
    </form>
    {{/each}}
    </tr>

</tbody>
</div>
</div>
</table></center>

```

O código é uma lista de itens que estarão no cookie, a única diferença do listar da pasta de produtos é que nos teremos uma opção de remover ao invés de adicionar.

```

Código Produto:<strong> {{this.id}}</strong> - <strong>{{this.nome}}</strong>
<a href="/carrinho/remover/{{this.id}}"><img

```

Vamos agora no listar.handlebars da pasta views/produtos e incluir o caminho e os objetos que serão adicionados ao carrinho via cookie.

Na linha onde temos o <a href="/carrinho/Adicionar" faremos a inclusão de /{{this.id\_produto}}/{{this.nome}} ficando o código assim:

```

<a href="/carrinho/Adicionar/{{this.id_produto}}/{{this.nome}}">

```

```
cod. Produto:{{this.id_produto}}<br>  
<a href="/carrinho/Adicionar/{{this.id_produto}}/{{this.nome}}"><img  
src="https://lh3.googleusercontent.com/pw/AP1GczN10lj86BUpc9ybef4ezxS__8o7qX1M
```

Vamos testar e pronto, finalizamos nossa aplicação.

Agora é com você aproveite para melhorar nossa aplicação, tenho certeza que ficará ótima com suas melhorias.

Faça alterações no código, refaça com novas atualizações.

Agradeço o tempo que ficamos juntos e espero ter contribuído um pouco com seu aprendizado.

*Nos vemos por aí.*

*Prof. Antonio Izo Júnior*