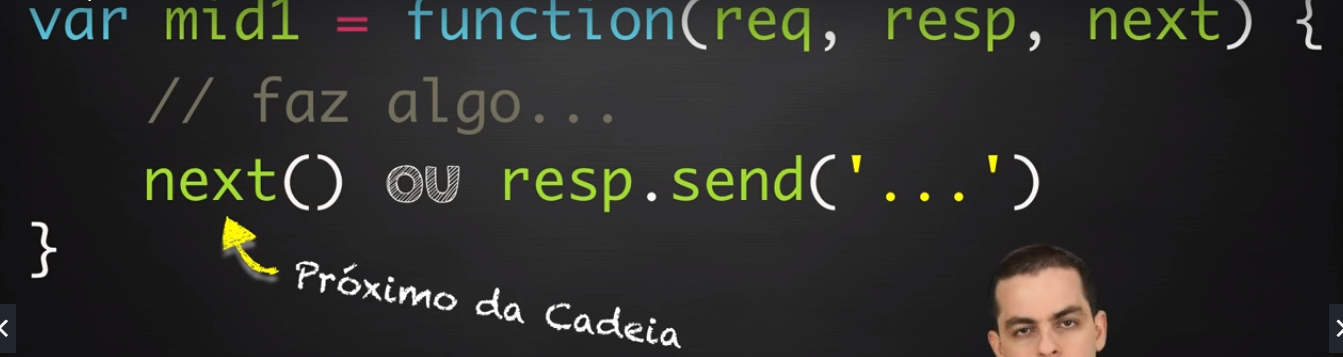
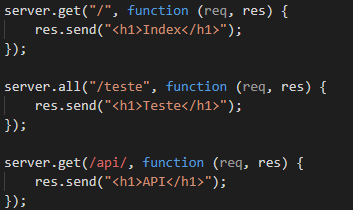
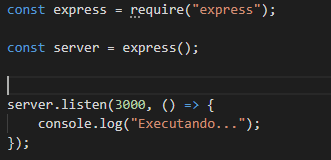
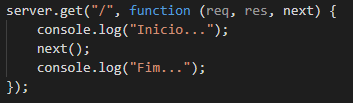
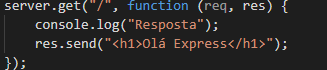
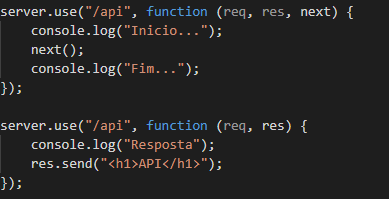
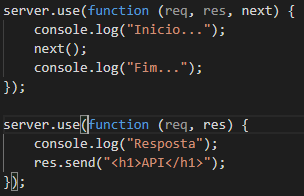
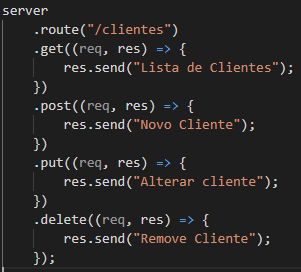
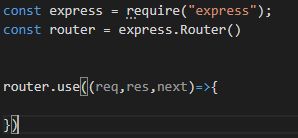
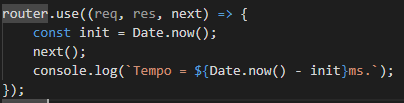
**Express**

* Express é no modo Chain of Responsability



* Quando uma url é inserida, o express irá chamar alguma função (middleware)
* **Configuração e Mapeamento de Rotas:**
  + Instalar um novo npm init
  + Npm i --save express (npm i é para instalar pelo package manager do node, o --save é para salvarmos a dependência em package.json)
  +  Instanciamos o express na nossa aplicação
  + Colocamos um middleware (função) quando for colocado uma url (no nosso caso ‘/’ que é a raiz da aplicação, ou seja, a página que vai iniciar)
  + Iremos inserir a aplicação em uma porta (3000) e chamarmos uma função assim que iniciarmos (no nosso caso, só colocamos um indicativo que está funcionando)
  + Temos várias formas de inserirmos umas middleware para uma url, podemos usar método POST, método GET, método PUT. Observe que no último get, colocamos uma expressão fora de “”, pois se caso colocarmos qualquer url que contenha “api” ele vai automaticamente para a função denominada para /api/ (exemplo: localhost:3000/api/teste ou localhost:3000/minha\_api)
* **Cadeia de Middlewares:**
  + Inicialização padrão de server
  +  Iremos criar um get na url principal, e iremos criar uma função com o parâmetro next (para chamarmos uma outra middleware
  +  Essa é a segunda middleware a ser chamada (em next() da middleware anterior)
  + Então a sequência é: Middleware 1 🡪 “Inicio...” 🡪 Entra na Middleware 2 🡪 “Resposta” 🡪 Cria o cabeçalho “Olá Express”🡪 Volta pra Middleware 1 🡪 “Fim...”
* **Método USE**
  + Foi utilizado o mesmo esquema utilizado em “Cadeia de Middlewares”, porém ao invés de usarmos get, usamos Use, e atribuímos para a url “/api”
  + O USE além de pegar diferentes métodos (GET, POST, PUT) para usar os middlewares, ele pega a possibilidade de realizar a middleware caso for inserido uma url do tipo localhost:3000/api/blablabla (diferentemente do GET, que se refere exatamente a url chamada). Porém ele tem que começar por /api, se a url for localhost:3000/teste/api ou localhost:3000/aapi NÃO VAI FUNCIONAR
  + Caso eu tirar a url a qual eu quero aplicar uma middleware, e só deixar a função, a middleware será inserida para QUALQUER url que eu insira
* **Método Route**
  + Será utilizada a mesma estrutura de Cadeia de middlewares para inicialização e roteamento
  + Podemos encadear para diferentes métodos uma mesma url. Se caso eu entrar com o método get na url localhost:3000/clientes, ele irá me retornar “Lista de Clientes”
* **Método Express-Route**
  + O Router é tipo uma lista de funções para determinados métodos. Por exemplo: “Eu quero um Router utilizado só para a API de cadastro de clientes”
  + Vários Routers podem ser criados
  +  Estrutura básica
  + Para todo método permitido por Use ele irá pegar o tempo da requisição
  + Iremos criar um middleware para o método Get, que irá pegar a url que tenha /produtos/”algum id” – Sendo “algum id” qualquer coisa, pode ser um número, uma string, etc
  + O método irá pegar o id passado na url, e passar em forma de json para o webserver (vai retornar um json)
  + Para exportarmos esse router para uso externo, iremos colocar: module.exports = router
  + No arquivo que iremos importar o router, iremos importa-lo da seguinte forma
  + Estamos colocando router como uma middleware iniciada com a url /api. Então para cadastrarmos um produto, temos de colocar a url localhost:3000/api/produtos/”idProduto”