* criar controller para a sessão de autenticação
* criar uma classe para o controller
* dentro da classe criar um método assíncrono de create
* desestruturar email e password do request.body e return o response
* exportar o controller
* criar uma rota para a sessão
* importar o Router
* importar o controller da sessão de autenticação
* instanciar o controller de autenticação
* iniciar o Router()
* criar uma requisição post com endereço de origem e com o método create do controller
* exportar o router
* no index das rotas
* importar a rota de autenticação
* no insomnia
* criar uma pasta sessions
* criar requisição post
* adicionar o endereço localhost:3000/sessions
* selecionar json
* criar e-mail e senha

VALIDANDO USUARIO

* no controller
* importar a conexão com o bd
* importar AppError
* recuperar o email de dentro do db(utilizar lógicas de pegar dados do bd)
* validar se usuario nao existe

VALIDANDO SENHA

* no controller
* importar o compare do bcrypt
* recuperar a senha de dentro do db(utilizar compare)
* validar se senha esta certa

GERANDO TOKEN

* instalar jsonwebtoken via npm install jsonwebtoken
* criar uma pasta configs e dentro um arquivo chamado auth.js
* dentro de auth.js
* criar um objeto já exportado contendo

{

jwt:{

secret:”default”,

expiresIn:”1d”,

}

}

* no controller
* importar o auth.js
* importar sign de jsonwebtoken
* desestruturar o secret e expiresIn de auth.js
* criar token, token = sign({},secret,{

subject:String(user.id),

expiresIn})

* adicionar user e token no return response

GUARDAR TOKEN NO INSOMNIA

* clicar em New Environment
* clicar em manage environment
* clicar em base environment
* criar variável “USER\_TOKEN”
* como valor da variável abrir aspas duplas “”
* dentro das aspas digite response, será aberta uma lista, selecione body attribute
* em request selecione o post da session
* em filter(jsonpath or xpath) digite $.token
* em trigger behavior selecione always resend request when needed

MIDDLEWARE DE AUTENTICAÇÃO

* criar pasta middleware em src
* dentro da pasta criar arquivo ensureAuthentication.js
* dentro de ensureAuthentication
* importar o verify de dentro de jsonwentoken
* importar AppError
* importar authConfig
* criar função com mesmo nome do arquivo, recebe como parâmetro request response e next. Recuperar o header da requisição e o authorization
* verificar se o token não existe, tratar o erro
* acessar através de um vetor o que estiver dentro do header
* dentro de um try-catch usar o verify passando o token e o secret do authConfig dentro de um objeto desestruturado passando um alias de user\_id.
* criar uma propriedade request.user como objeto, criar uma chave id que recebe o user\_id em formato de número
* retornar o next
* no catch retornar algum erro
* exportar a função

UTILIZANDO O MIDDLEWARE DE AUTENTICAÇÃO

* no arquivo users.routes.js
* importar o middleware de autenticação
* colocar o middleware somente nas rotas que precisar
* o middleware deve ser inserido antes da funcao que for fazer(update por exemplo)
* quando tem middleware não é mais necessário passar o id do usuário como dado(na requisição)
* na função(update por exemplo) não é mais necessário recuperar o id pelos parâmetro, deve-se passar o user que foi criado no middleware e pegar o id
* na função substituir o id do usuário pelo user\_id
* inserir o token no update do insomnia
* fazer esse procedimento para todas as rotas e funcoes que precisar na sua aplicação