

Curso Técnico Superior Profissional em Tecnologias e Programação de Sistemas de Informação

Unidade Curricular: Desenvolvimento Web – Front-End

1.º Ano/2.º Semestre

Docente: Marco Miguel Olival Olim

Data 03/05/2019

ESTE EXERCÍCIO ABORDA O PROCESSO DE AUTENTICAÇÃO DE UM SPA EM VUEJS COM UM BACK-END NODEJS

A abordagem mais utilizada ao implementar um sistema de autenticação para uma *Single Page Application* é a *cross-domain authentication* atendendo a que uma aplicação atual pode ter múltiplas fontes de dados. Assim sendo, o token de autenticação tem de acompanhar o header de cada pedido. No caso da aplicação que estamos a desenvolver, o back-end e o front-end está no mesmo domínio (*local* ou *same-domain authentication*) e por isso controlamos todo o fluxo de autenticação e autorização, não necessitando assim da estratégia anterior que acrescenta muito mais complexidade a todo o processo. Iremos utilizar o Passport.js para gerir a autenticação no servidor (que irá persistir com uma cookie), sendo este implementado em NodeJS e Express.js. Depois de criar o ficheiro de configuração **package.json** instale com **npm install**

```
{
  "name": "auth-backend",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "author": "",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.19.0",
    "cookie-session": "^1.3.3",
    "express": "^4.16.4",
    "passport": "^0.4.0",
    "passport-local": "^1.0.0"
  }
}
```

Crie agora o ficheiro index.js para criar um servidor em nodeJS

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

app.listen(3000, () => {
  console.log("port: 3000")
})
```

Execute com o nodeJS para verificar se o servidor está operacional

```
C:\Users\olimm\Desktop\vue\cookieauth>node index.js
port: 3000
```

Cofinanciado por:

Após a confirmação que o servidor está a funcionar na porta 3000 vamos agora criar uma pasta **wwwroot** (ou outro nome que considere apropriado) onde irão ficar os ficheiros do website

```
const express = require('express')
const app = express()
const publicRoot = './wwwroot/'
app.use(express.static(publicRoot))

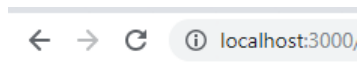
app.get("/", (req, res, next) => {
  res.sendFile("index.html", { root: publicRoot })
})

app.listen(3000, () => {
  console.log("port: 3000")
})
```

Crie um ficheiro de html para colocar nessa pasta

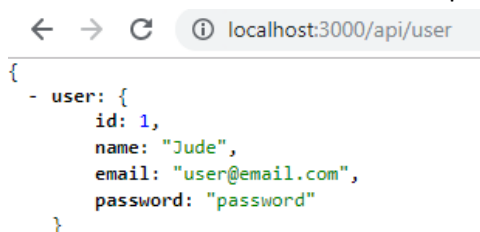
```
<> index.html ●
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5    <meta charset="UTF-8">
6    <meta name="viewport" content="width=device-width, initial-scale=1.0">
7    <meta http-equiv="X-UA-Compatible" content="ie=edge">
8    <title>Document</title>
9  </head>
10
11 <body>
12   <h1>users</h1>
13 </body>
14
15 </html>
```

Teste agora no browser. Atenção: após qualquer alteração deve voltar a executar **node server.js**



users

Vamos agora desenvolver a API. Esta deverá receber um pedido do browser, consultar a base de dados para obter essa informação e resolver o pedido com o JSON pretendido. Neste momento ainda não temos uma base de dados funcional, pelo que vamos emular esta parte com a declaração de uma variável **users**. Neste momento os users serão o único endpoint da API a ser disponibilizado em <http://localhost:3000/api/user>



Cofinanciado por:



Deverá ter assim o index.js:

```
const express = require('express')
const app = express()
const publicRoot = './wwwroot/'
app.use(express.static(publicRoot))

let users = [{
  id: 1,
  name: "Jude",
  email: "user@email.com",
  password: "password"
},
{
  id: 2,
  name: "Emma",
  email: "emma@email.com",
  password: "password2"
},
]

app.get("/", (req, res, next) => {
  res.sendFile("index.html", { root: publicRoot })
})
app.get("/api/user", (req, res) => {
  console.log(users)
  res.send(users)
})
app.listen(3000, () => {
  console.log("port: 3000")
})
```

E a consola deverá retornar após o pedido do browser:

```
C:\Users\olimm\Desktop\vue\cookieauth>node index.js
port: 3000
[ { id: 2,
  name: 'Emma',
  email: 'emma@email.com',
  password: 'password2' },
```

Vamos agora implementar a autenticação. Como já foi referido, será utilizado o Passport.js que irá criar e gerir uma sessão para cada utilizador. Utilizaremos a *same-domain authentication* pelo que necessitamos do add-on *passport-local.js* para implementar esta estratégia.

```
const app = express()
const cookieSession = require('cookie-session')
const bodyParser = require('body-parser')
|
const passport = require('passport')
const LocalStrategy = require('passport-local').Strategy
```

Cofinanciado por:



Estas sessões serão geridas por cookies que irão persistir por 24h no browser

```
app.use(bodyParser.json())
app.use(cookieSession({
  name: 'mysession',
  keys: ['vueauthrandomkey'],
  maxAge: 24 * 60 * 60 * 1000 // 24 horas
}))

app.use(passport.initialize());
app.use(passport.session());
```

Temos agora que adaptar a nossa API para retornar o utilizador autenticado. Para proteger este endpoint de acessos não autorizados (401) aplicamos um filtro (middleware) para verificar se a sessão é válida antes de continuar com a resolução do pedido

```
const authMiddleware = (req, res, next) => {
  if (!req.isAuthenticated()) {
    res.status(401).send('Não está autenticado')
  } else {
    return next()
  }
}

app.get("/api/user", authMiddleware, (req, res) => {
  let user = users.find((user) => {
    return user.id === req.session.passport.user
  })
  console.log([user, req.session])
  res.send({ user: user })
})
```

Para gerir as credenciais com o Passport.js temos de estabelecer os endpoints de login (e também de logout) de forma a poder fornecer as cookies com a sessão e invalidá-las no procedimento de logout.

```
app.post("/api/login", (req, res, next) => {
  passport.authenticate('local', (err, user, info) => {
    if (err) {
      return next(err);
    }

    if (!user) {
      return res.status(400).send([user, "Não é possível realizar o Login", info])
    }

    req.login(user, (err) => {
      res.send("Logged in")
    })
  })(req, res, next)
})

app.get('/api/logout', function(req, res) {
  req.logout();
  console.log("logged out")
  return res.send();
});
```

Cofinanciado por:



Por fim configuramos o Passport.js para gerir os pedidos dos diversos endpoints da API. Note que para efetuarmos quaisquer alterações no processo antes de gravar na sessão temos de serializar a resposta. A título de exemplo vamos juntar o id do utilizador. Poderá usar este processo para a encriptação da password.

```
passport.use(new LocalStrategy({
  usernameField: 'email',
  passwordField: 'password'
}),
(username, password, done) => {
  let user = users.find((user) => {
    return user.email === username && user.password === password
  })

  if (user) {
    done(null, user)
  } else {
    done(null, false, { message: 'Password ou Email inválidos' })
  }
})

passport.serializeUser((user, done) => {
  done(null, user.id)
})

passport.deserializeUser((id, done) => {
  let user = users.find((user) => {
    return user.id === id
  })

  done(null, user)
})

app.listen(3000, () => {
```

Agora vamos desenvolver o front-end em VueJS. Pode utilizar o projeto existente ou criar um novo numa nova pasta **vue** (recomenda-se que crie o projeto com o Vue CLI como nas fichas anteriores), edite o package.json, para adicionar o **vue-cookies** (além do **axios**), executando depois o **npm install**

```
{
  "name": "vue",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "serve": "vue-cli-service serve",
    "build": "vue-cli-service build"
  },
  "dependencies": {
    "vue": "^2.6.10",
    "vue-cookies": "^1.5.13",
    "vue-router": "^3.0.3"
  },
  "devDependencies": {
    "@vue/cli-service": "^3.7.0",
```

Cofinanciado por:

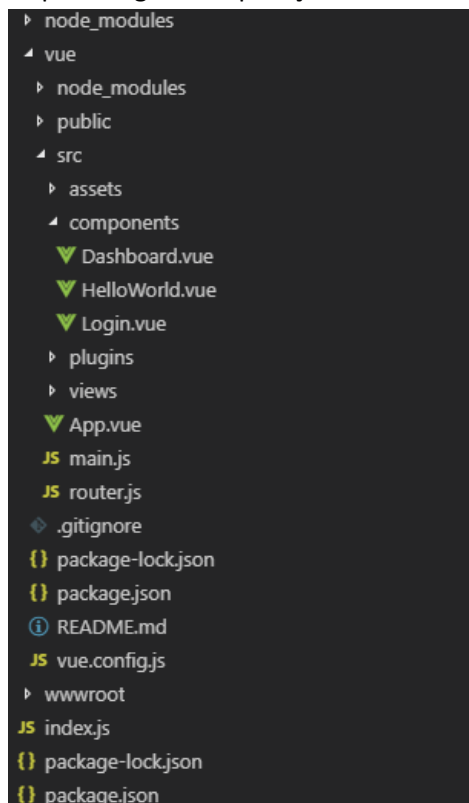


```

    "axios": "^0.18.0",
    "vue-cli-plugin-axios": "0.0.4",
    "vue-template-compiler": "^2.5.21"
  },
  "postcss": {
    "plugins": {
      "autoprefixer": {}
    }
  },
  "browserslist": [
    "> 1%",
    "last 2 versions"
  ]
}

```

Depois de gerar a aplicação deverá ter a seguinte estrutura, à qual irá adicionar o dashboard.vue e login.vue



Vamos primeiro editar o **main.js** para configurar a utilização de cookies

```

JS main.js
1  import Vue from 'vue'
2  import './plugins/axios'
3  import App from './App.vue'
4  import router from './router'
5  import VueCookies from "vue-cookies"
6  Vue.use(VueCookies)
7
8  Vue.config.productionTip = false
9
10 new Vue({
11   router,
12   render: function(h) { return h(App) }
13 }).$mount('#app')

```

Cofinanciado por:



Depois, caso não esteja a utilizar o projeto existente, modificamos o App.vue para podermos navegar na app

```
▼ Login.vue ▼ Dashboard.vue ▼ App.vue ×
1  <template>
2    <div id="app">
3
4      <div>
5        <router-link :to="{ name: 'Dashboard'}" class="button--green">Dashboard</router-link>
6        <router-link :to="{ name: 'Login'}" class="button--green">Login</router-link>
7        <a href="#" @click="logout" class="button--green">Logout</a>
8      </div>
9      
10     <router-view/>
11   </div>
12 </template>
13
14 <script>
15 import router from "./router"
16 export default {
17   name: "App",
18   methods: {
19     logout(e) {
20       axios.get("/api/logout")
21         .then(() => {
22           console.log("Logged out")
23           router.push("/")
24         })
25         .catch((errors) => {
26           console.log(errors)
27         })
28     }
29   }
30 }
31 </script>
32
33 <style>
34 #app {
35   font-family: 'Avenir', Helvetica, Arial, sans-serif;
36   -webkit-font-smoothing: antialiased;
37   -moz-osx-font-smoothing: grayscale;
38   text-align: center;
39   color: #2c3e50;
40   margin-top: 60px;
41 }
42 .button--green
43 {
44   display: inline-block;
45   border-radius: 4px;
46   border: 1px solid #3b8070;
47   color: #3b8070;
48   text-decoration: none;
49   padding: 10px 30px;
50   margin: 5px;
51 }
52 .button--green:hover
53 {
54   color: #fff;
55   background-color: #3b8070;
56 }
57 </style>
```

Cofinanciado por:



O componente de login, a ser criado em **components/Login.vue**, contém um formulário para enviar com o axios para o endpoint `/api/login` um objeto com o email e password introduzidos, redirecionando para o *dashboard* após completar o pedido com sucesso.

```
▼ Login.vue x ▼ Dashboard.vue ▼ App.vue
1  <template>
2    <div>
3      <h2>Login</h2>
4      <form v-on:submit="login">
5        <input type="text" name="email" placeholder="email" class="input-control"/><br>
6        <input type="password" name="password" placeholder="password" class="input-control"/><br>
7        <input type="submit" value="Login" class="button--green"/>
8      </form>
9    </div>
10 </template>
11
12 <script>
13 import router from "../router"
14 import axios from "axios"
15 export default {
16   name: "Login",
17   methods: {
18     login: (e) => {
19       e.preventDefault()
20       let email = e.target.elements.email.value
21       let password = e.target.elements.password.value
22       let login = () => {
23         let data = {
24           email: email,
25           password: password
26         }
27         axios.post("/api/login", data)
28           .then((response) => {
29             console.log("Logged in")
30             router.push("/dashboard")
31           })
32           .catch((errors) => {
33             console.log(errors)
34           })
35       }
36       login()
37     }
38   }
39 }
40 </script>
41 <style scoped>
42 .input-control {
43   margin: 10px 0;
44   font: inherit;
45   border: 1px solid #ccc;
46   border-radius: 4px;
47   padding: 5px;
48 }
49 </style>
```

Cofinanciado por:



O dashboard.vue é apenas um componente que pretende ilustrar uma implementação genérica de qualquer view do projeto que precise de autenticação para ser visualizada. Caso contrário redireciona para a página principal (recomenda-se até que redirecione para uma página específica de login)

```
▼ Login.vue    ▼ Dashboard.vue ●    ▼ App.vue
1  <template>
2    <div>
3      <h2>Dashboard</h2>
4      <p>Name: {{ user.name }}</p>
5    </div>
6  </template>
7
8  <script>
9    import axios from "axios"
10   import router from "../router"
11   export default {
12     name: "Login",
13     data () {
14       return {
15         user: {
16           name: "Offline"
17         }
18       }
19     },
20     methods: {
21       getUserData: function () {
22         let self = this
23         axios.get("/api/user")
24           .then((response) => {
25             console.log(response)
26             self.$set(this, "user", response.data.user)
27           })
28           .catch((errors) => {
29             console.log(errors)
30             router.push("/")
31           })
32       }
33     },
34     mounted () {
35       this.getUserData()
36     }
37   }
38 </script>
```

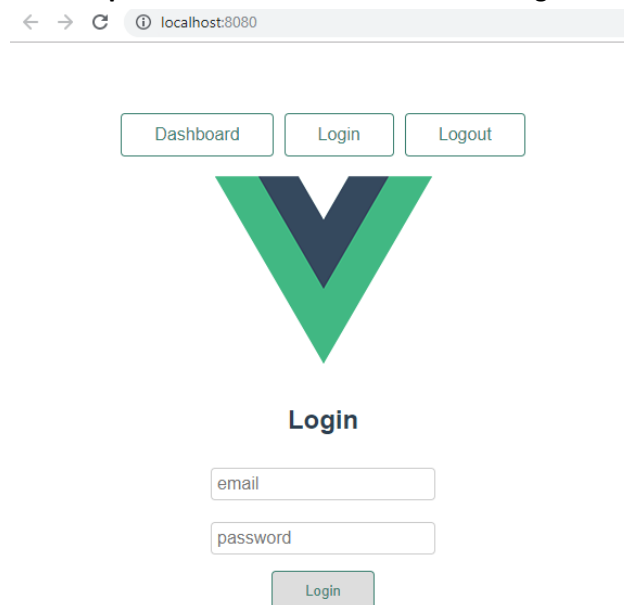
Para concluir, configuramos as novas rotas:

```
EXPLORER
OPEN EDITORS
x JS router.js vue/src
COOKIEAUTH
node_modules
vue
node_modules
public
src
assets
components
▼ Dashboard.vue
▼ HelloWorld.vue
▼ Login.vue
plugins
views
▼ App.vue
JS main.js
JS router.js
.gitignore

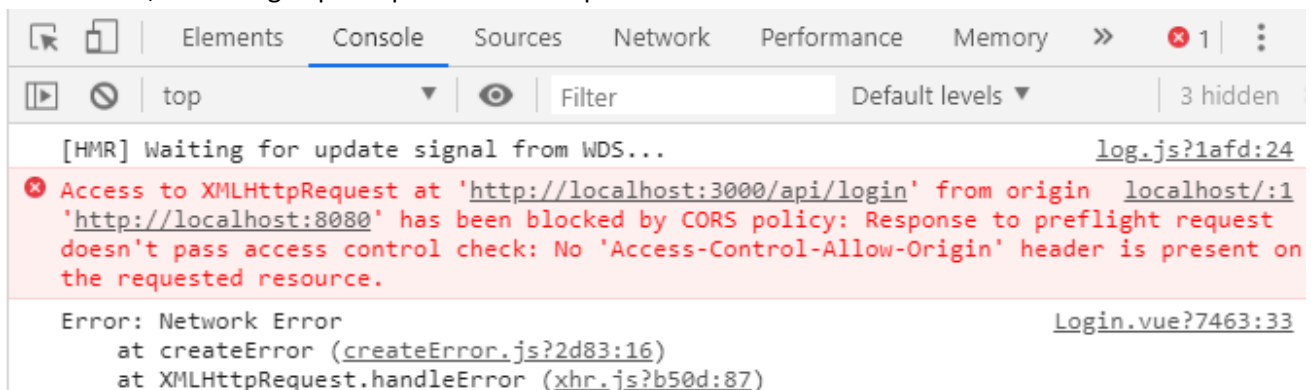
JS router.js x
1  import Vue from 'vue'
2  import Router from 'vue-router'
3  import Login from "@components/Login"
4  import Dashboard from "@components/Dashboard"
5
6  Vue.use(Router)
7
8  export default new Router({
9    mode: 'history',
10   base: process.env.BASE_URL,
11   routes: [{
12     path: '/',
13     name: 'Login',
14     component: Login
15   },
16   {
17     path: "/dashboard",
18     name: "Dashboard",
19     component: Dashboard
20   }
21 ]
22 })
```

Cofinanciado por:

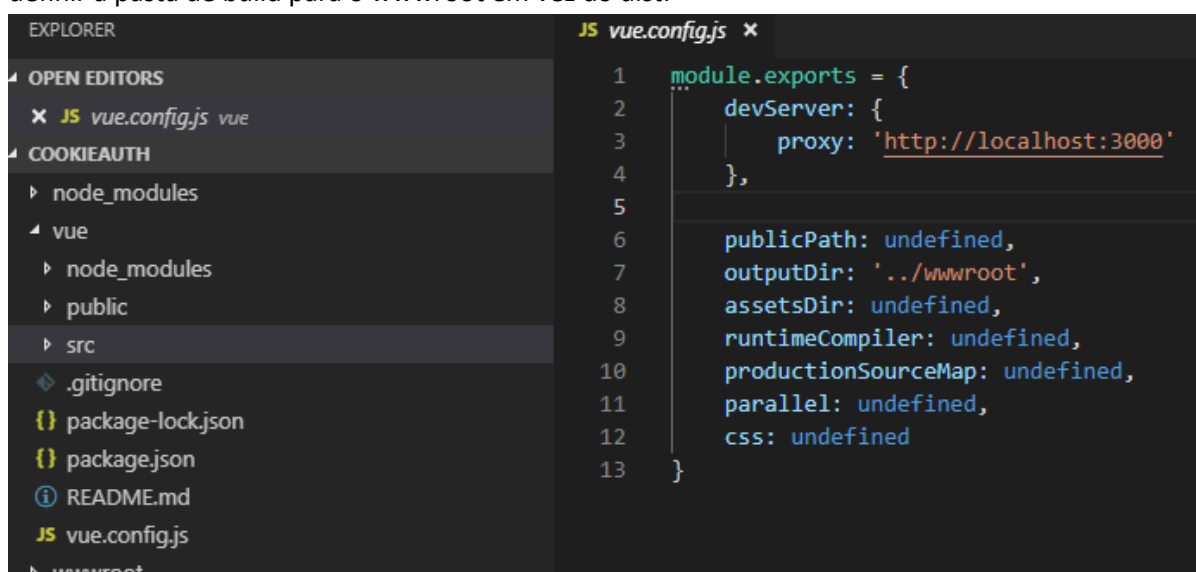
Pode agora executar o projeto com **npm run serve** ou utilize o ambiente gráfico com **vue ui**.



Apesar de não ter tido erros na compilação, quando tenta efetuar o login terá um erro de CORS (Cross-Origin Resource Sharing), visível na consola do browser. Este é expectável quando tentar aceder à API porque, lembre-se, a estratégia que implementamos é para *same-domain authentication*.

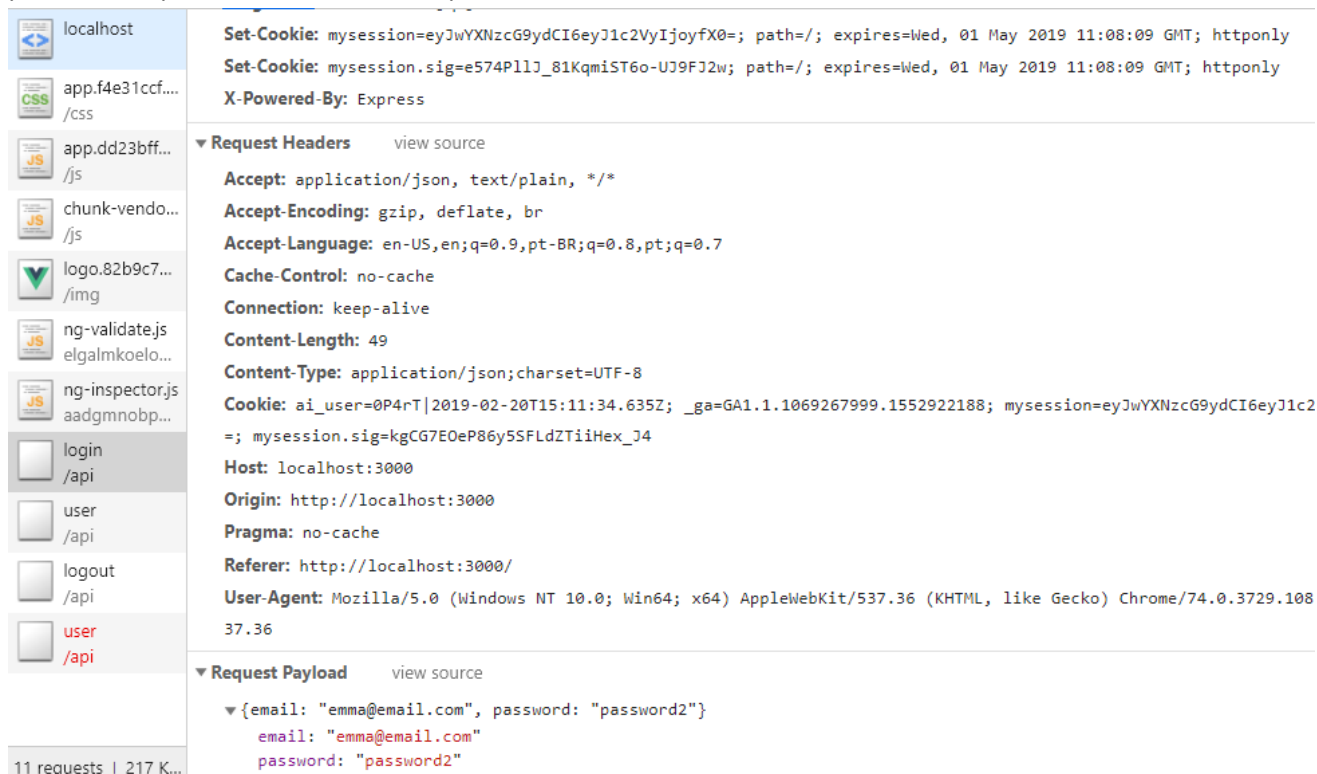


Precisa, por isso, de adicionar o ficheiro **vue.config.js** para configurar (por proxy) o servidor de desenvolvimento do vue para o mesmo porto do servidor de node (3000) de forma a prevenir erros de CORS. Para mais informações consultar <https://cli.vuejs.org/config/#devserver-proxy>. Neste ficheiro também pode definir a pasta de build para o wwwroot em vez do dist.



Cofinanciado por:

Após reiniciar o projeto de forma a que o `vue.config.js` seja executado, já deverá ser redirecionado para o dashboard após o login. Repare na consola do browser que no payload do pedido é enviado o login e password e que é criada uma cookie para a sessão.



localhost

app.f4e31ccf... /css

app.dd23bff... /js

chunk-vendo... /js

logo.82b9c7... /img

ng-validate.js elgalmkoelo...

ng-inspector.js aadgmnoBP...

login /api

user /api

logout /api

user /api

11 requests | 217 K...

Set-Cookie: mysession=eyJwYXNzcG9ydCI6eyJ1c2VyIjoyfX0=; path=/; expires=Wed, 01 May 2019 11:08:09 GMT; httponly

Set-Cookie: mysession.sig=e574P1lJ_81KqmiST6o-UJ9FJ2w; path=/; expires=Wed, 01 May 2019 11:08:09 GMT; httponly

X-Powered-By: Express

Request Headers view source

Accept: application/json, text/plain, */*

Accept-Encoding: gzip, deflate, br

Accept-Language: en-US,en;q=0.9,pt-BR;q=0.8,pt;q=0.7

Cache-Control: no-cache

Connection: keep-alive

Content-Length: 49

Content-Type: application/json; charset=UTF-8

Cookie: ai_user=0P4rT|2019-02-20T15:11:34.635Z; _ga=GA1.1.1069267999.1552922188; mysession=eyJwYXNzcG9ydCI6eyJ1c2VyIjoyfX0=; mysession.sig=kgCG7E0eP86y5SFLdZTiiHex_J4

Host: localhost:3000

Origin: http://localhost:3000

Pragma: no-cache

Referer: http://localhost:3000/

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.108 37.36

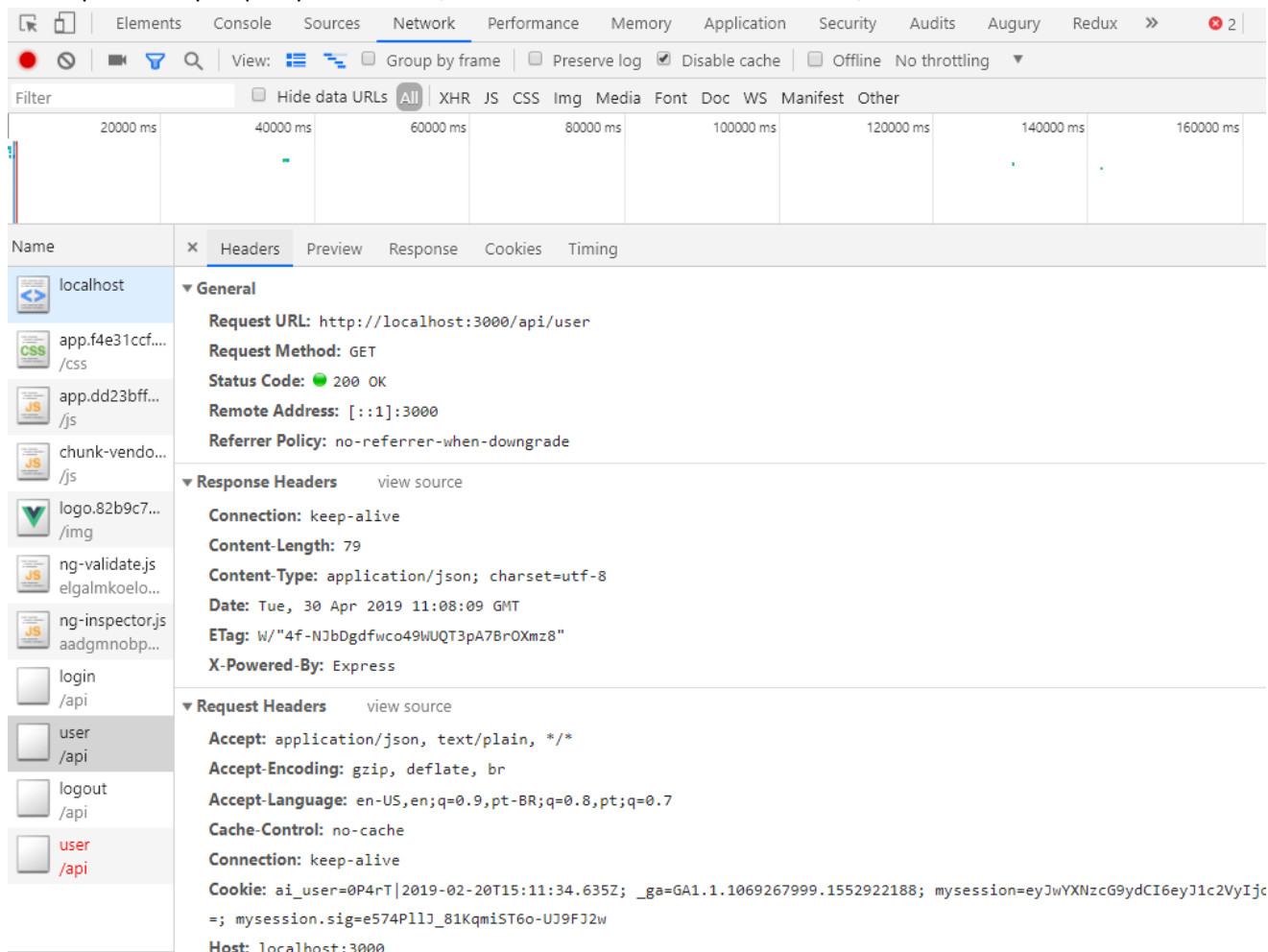
Request Payload view source

{email: "emma@email.com", password: "password2"}

email: "emma@email.com"

password: "password2"

Na resposta de qualquer pedido à API, essa cookie é adicionada ao header, identificando o user



localhost

app.f4e31ccf... /css

app.dd23bff... /js

chunk-vendo... /js

logo.82b9c7... /img

ng-validate.js elgalmkoelo...

ng-inspector.js aadgmnoBP...

login /api

user /api

logout /api

user /api

Name

Headers Preview Response Cookies Timing

General

Request URL: http://localhost:3000/api/user

Request Method: GET

Status Code: 200 OK

Remote Address: [::1]:3000

Referrer Policy: no-referrer-when-downgrade

Response Headers view source

Connection: keep-alive

Content-Length: 79

Content-Type: application/json; charset=utf-8

Date: Tue, 30 Apr 2019 11:08:09 GMT

ETag: W/"4f-NJbDgdfwco49WUQT3pA7BrOXmz8"

X-Powered-By: Express

Request Headers view source

Accept: application/json, text/plain, */*

Accept-Encoding: gzip, deflate, br

Accept-Language: en-US,en;q=0.9,pt-BR;q=0.8,pt;q=0.7

Cache-Control: no-cache

Connection: keep-alive

Cookie: ai_user=0P4rT|2019-02-20T15:11:34.635Z; _ga=GA1.1.1069267999.1552922188; mysession=eyJwYXNzcG9ydCI6eyJ1c2VyIjoyfX0=; mysession.sig=e574P1lJ_81KqmiST6o-UJ9FJ2w

Host: localhost:3000

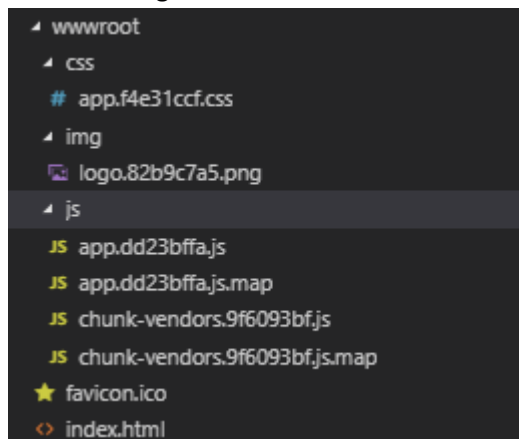
Cofinanciado por:



A aplicação está pronta para fazer o build para produção

The screenshot shows the Vue CLI interface with the 'build' task completed. The left sidebar lists 'Dashboard', 'Plugins', 'Dependencies', 'Configuration', and 'Tasks'. The 'Tasks' section shows 'serve' (Terminated), 'build' (Done), and 'inspect' (Inspect the resolved webpa...). The main area displays the 'build' task details, including a 'Run task' button, 'Parameters', and 'Output'. The 'Dashboard' section shows 'Status: Success', 'Errors: 0', 'Warnings: 0', and a large green checkmark. Below this, it lists 'Assets: 122.8kB (Parsed)', 'Modules: 109.3kB (Parsed)', and 'Dependencies: 105.9kB 96.87%'. The 'Speed stats' section shows various performance metrics for different network types and devices.

Deverá ter agora no wwwroot os ficheiros de produção numa estrutura semelhante à abaixo apresentada



Caso este build não grave os ficheiros para o wwwroot, edite o output directory em Parameters

The screenshot shows the 'Parameters' dialog box in the Vue CLI interface. It includes a 'Modern mode' toggle, a 'Specify env mode' dropdown set to 'production (Default)', an 'Output directory' text field containing './wwwroot', a 'Build target' dropdown set to 'Web app (Default)', a 'Name for library or web-component mode' text field, and a 'Watch for changes' toggle. A 'Save' button is at the bottom.


Cofinanciado por:



A aplicação já funciona autonomamente com o servidor de nodeJS (no porto 3000)

← → ↻ ⓘ localhost:3000

[Dashboard](#) [Login](#) [Logout](#)



Login

[Login](#)

Aplique agora este processo de autenticação ao projeto, colocando o login num componente próprio devidamente formatado com o Vuetify (recomenda-se a utilização de v-text-field dentro de v-card)

← → ↻ ⓘ localhost:8080/signin

[VueShare](#) [POSTS](#) [SIGN IN](#) [SIGN UP](#)

Welcome Back!

[SIGNIN](#)

Don't have an account? [Signup](#)

Cofinanciado por:

