Aplicações para a Internet II

NodeJS e React

2020/2021

< JWT - Autenticação por tokens />

- **JSON Web Tokens** (JWT) – *Padrão RFC-7519*
  - Autenticação baseado em tokens;
  - Para cada solicitação (*request*) um token é passado para autenticação;
  - Um token é uma string utilizada pelo cliente e pelo servidor para autenticar de uma forma fácil;
  - O token é gerado a partir de uma chave segura definida por nós;
  - Esse token é passado para o cliente. Sempre que o cliente envia esse token junto com uma solicitação, o servidor valida-o e devolve a resposta.

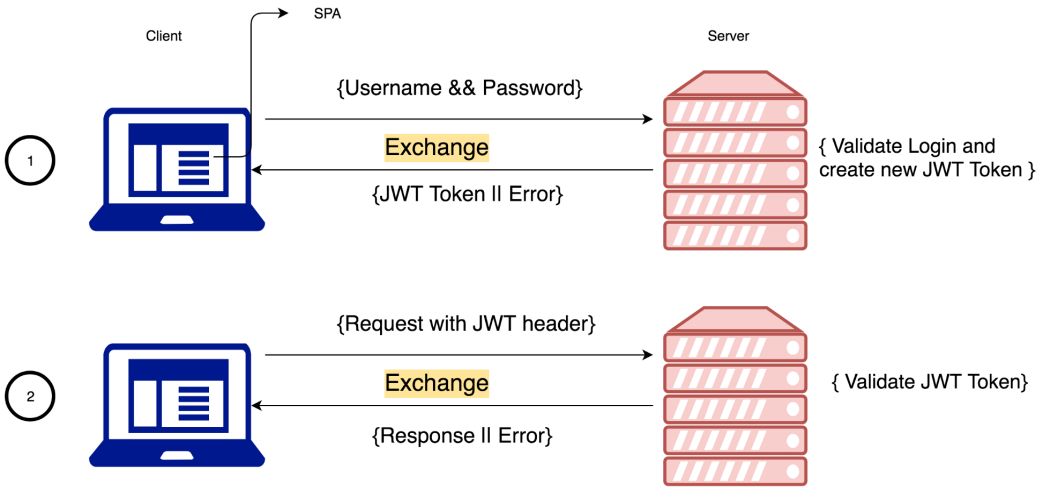

Imagem retirada do artigo
https://medium.com/dev-bits/a-guide-for-adding-jwt-token-based-authentication-to-your-single-page-nodejs-applications-c403f7cf04f4

< JWT - Autenticação por tokens />

- **O que é um Token da Web JSON e por que precisamos dele?**
- Um JSON Web Token (JWT) é uma maneira segura, compacta e independente de transmitir informações entre várias *endpoints,* na forma de um objeto JSON.
- Estrutura do JWT:
  - Um JSON Web Token consiste em três partes separadas por um ".". Elas são:
    - Header: o tipo do token e o algoritmo de hash que está a ser usado.
    - Payload: onde as informações que queremos enviar são armazenadas.
    - Signature: usada para verificar se a mensagem não foi alterada antes de chegar ao destino.

Além da autenticação, o JSON Web Token é uma maneira excelente e segura de transmitir dados entre várias partes. O facto das JWTs terem assinaturas permite que todos possam identificar os remetentes, necessitando apenas da chave correta.

Imagem retirada do artigo
https://blog.bitsrc.io/understanding-json-web-token-authentication-a1febf0e15

< Node.js (backend) />

- **Que módulos NPM é necessário instalar?**
  - **JSON Web Tokens** (*jsonwebtoken*) - https://www.npmjs.com/package/jsonwebtoken
  - **Bcrypt** (*bcrypt*) - https://www.npmjs.com/package/bcrypt

< Node.js (backend) />

- No **backend** é necessário começar por criar o ficheiro '**config.js**' que conterá a nossa chave privada de encriptação que será usada para gerar o *token*;
- **Crie o ficheiro src/config.js:**

```
module.exports = {
  jwtSecret: 'chavesecreta'
};
```

- De seguida é necessário criar o ficheiro '**middleware.js**' que tem como função verificar o *token* de autenticação recebido em cada solicitação (*req*) e, deste modo, decidir se o pedido chega ou não à rota.

```
const jwt = require('jsonwebtoken'); //módulo NPM
const config = require('./config.js'); //ficheiro de configuração

let checkToken = (req, res, next) => {
let token = req.headers['x-access-token'] ||
            req.headers['authorization'];
```

< Node.js (backend) – middleware />

```
if (token.startsWith('Bearer ')) {
  token = token.slice(7, token.length); //remove a palavra 'Bearer '
}

if (token) {
  jwt.verify(token, config.jwtSecret, (err, decoded) => {
      if (err) {
        return res.json({
          success: false,
          message: 'O token não é válido.'
        });
      } else {
        req.decoded = decoded;
        next();
      }
    });
  } else {
      return res.json({
        success: false,
        message: 'Token
              indisponível.'
      });
  }
};

module.exports = {
  checkToken: checkToken
}
```

< Node.js (backend) - Model />

- Os dados dos utilizadores ficarão registados numa tabela '**users**' com os campos ***id***, ***name***, ***email*** e ***password***;
- Na pasta '**models**' é necessário criar o ficheiro '**User.js**' com o seguinte código:

```
const Sequelize = require('sequelize');
const sequelize = require('./database');
const bcrypt = require('bcrypt'); //encripta a pass a guardar na BD

var User = sequelize.define('user', {
  id: {
    type: Sequelize.INTEGER,
    primaryKey: true,
    autoIncrement: true,
  },
  name: Sequelize.STRING,
  email: {
    type: Sequelize.STRING,
    allowNull: false,
    unique: true
  },
  password:{
    type: Sequelize.STRING,
    allowNull: false
  }
}, {
  timestamps: false,
});
```

< Node.js (backend) - Model />

```
User.beforeCreate((user, options) => {
    return bcrypt.hash(user.password, 10)
    .then(hash => {
        user.password = hash;
    })
    .catch(err => {
        throw new Error();
    });
});

module.exports = User;
```

< Node.js (backend) - Route />

- Na pasta '**routes**' é necessário criar o ficheiro '**userRoute.js**' com o seguinte código:

```
const express = require('express');
const router = express.Router();
const middleware = require('../middleware');

//importar os controladores
const userController = require('../controllers/userController')

router.get('/list', middleware.checkToken, userController.list);
router.post('/register',userController.register);
router.post('/login',userController.login);

module.exports = router;
```

< Node.js (backend) - Controller />

- Na pasta '**controllers**' é necessário criar o ficheiro '**userController.js**' com o seguinte código:

```
const jwt = require('jsonwebtoken');
const bcrypt = require('bcrypt');
const User = require('../model/User');
const sequelize = require('../model/database');
const config = require('../config');

const controllers = {}
sequelize.sync()

controllers.list = async (req, res) => {
    const data = await User.findAll()
    .then(function(data){
      return data;
    })
    .catch(error => {
      return error;
    });
    res.json({success: true,
              data: data});
}
```

< Node.js (backend) - Controller />

```
controllers.register = async (req,res) => {
  const { name, email, password } = req.body;
  const data = await User.create({
    name: name,
    email: email,
    password: password
  })
  .then(function(data){
    return data;
  })
  .catch(error =>{
    console.log("Erro: "+error);
    return error;
  })
```

```
res.status(200).json({
    success: true,
    message:"Registado",
    data: data
  });
}
```

< Node.js (backend) - Controller />

```
controllers.login = async (req,res) => {
  if (req.body.email && req.body.password) {
    var email = req.body.email;
    var password = req.body.password;
  }
  var user = await User.findOne({where: { email: email}})
  .then(function(data){
    return data;
  })
  .catch(error =>{
    console.log("Erro: "+error);
    return error;
  })
  if (password === null || typeof password === "undefined") {
    res.status(403).json({
      success: false,
      message: 'Campos em Branco'
    });
```

< Node.js (backend) - Controller />

```
} else {
   if (req.body.email && req.body.password && user) {
    const isMatch = bcrypt.compareSync(password, user.password);
    if (req.body.email === user.email && isMatch) {
      let token = jwt.sign({email: req.body.email}, config.secret,
                           {expiresIn: '1h' //expira em 1 hora
      });
      res.json({success: true, message: Autenticação realizada com
                                        sucesso!', token: token});
    } else {
      res.status(403).json({success: false, message: 'Dados de
                                          autenticação inválidos.'});
    }
  } else {
    res.status(400).json({success: false, message: 'Erro no processo de
                              autenticação. Tente de novo mais tarde.'});
  }
}} module.exports = controllers;
```

< Node.js (backend) – App />

- Por fim é necessário fazer pequenas alterações no ficheiro '**app.js**' com o intuito de operacionalizar o sistema de autenticação da aplicação;
- As alterações a efetuar são:
  - Importar a *middleware* para a aplicação;

```
(…)
const middleware = require('./middleware');
(…)
```

  - Implementar a rota de autenticação que inclui a função de login, registo, etc.;

```
(…)
app.use('/user', userRouters);
(…)
```

  - Adicionar a camada de segurança à rota dos funcionários com recurso à *middleware*;

```
(…)
app.use('/employee', middleware.checkToken, employeeRouters);
(…)
```

< Node.js (backend) – App />

```
const express = require("express");
const middleware = require("./middleware");
const app = express();
const cors=require("cors");
app.use(cors());
//Configurações
app.set("port", process.env.PORT || 3000);
//Middlewares
app.use(express.json());

//Rotas
const userRouters = require("./routes/userRoute");
app.use("/user", userRouters);

const employeeRouters = require("./routes/employeeRoute.js");
app.use("/employee", middleware.checkToken, employeeRouters);
```

```
app.use("/teste", (req, res) => {
  res.send("Rota TESTE.");
});

app.use("/", (req, res) => {
  res.send("Hello World");
});

app.listen(app.get("port"), () => {
  console.log("Start server on port " + app.get("port"));
});
```

< React.js (frontend) />

- No **frontend** vamos começar por criar um componente que fará a gestão do serviço de autenticação;
- Na pasta '**views**', crie o ficheiro '**auth.service.js**', com o seguinte código:

```
import axios from "axios";

class AuthService {
  login(email, password) {
    return axios
      .post("http://localhost:8000/user/login", {email, password})
      .then(res => {
        if (res.data.token) {
          localStorage.setItem("user", JSON.stringify(res.data));
        }
        return res.data;
      }, reason => { throw new Error('Utilizador Inválido');});
  }
  logout() { localStorage.removeItem("user"); }
  getCurrentUser() { return JSON.parse(localStorage.getItem('user'));}
} export default new AuthService();
```

< React.js (frontend) />

- Com o intuito de simplificar e automatizar as solicitações (*request*) ao **backend**, iremos implementar um componente que inclui automaticamente o utilizador (*user*) no *header* de todas as solicitações;
- Na pasta '**views**', crie o ficheiro '**auth-header.js**', com o seguinte código:

```
export default function authHeader() {
  const user = JSON.parse(localStorage.getItem('user'));

  if (user && user.token) {
    return { Authorization: 'Bearer ' + user.token };
  } else {
    return {};
  }
}
```

- No componente já existente de listar funcionários (*list.js*) iremos atualizar o código que faz a solicitação dos dados ao **backend**;

< React.js (frontend) - List />

- Na pasta '**views**', abra o ficheiro '**list.js**', e insira o seguinte código:

```
import authHeader from './auth-header';
(…)
 function LoadEmployee() {
    const url = "http://localhost:3000/employee/list";
    axios
      .get(url, { headers: authHeader() })
      .then((res) => {
        console.log(res);
        if (res.data.success) {
          const data = res.data.data;
          setdataEmployee(data);
        } else {
          alert("Error Web Service!");
        }
      })
      .catch((error) => {
        alert(error);
      });
 }(…)
```

< React.js (frontend) - List />

```
(…)
sendDelete(userId) {
  const baseUrl = "http://localhost:8000/employee/delete"
  axios.post(baseUrl, {headers: authHeader(), id: userId})
    .then(response =>{
      if (response.data.success) {
        Swal.fire('Deleted!', 'Your employee has been deleted.',
                  'success');
        LoadEmployee();
      }
    })
    .catch ( error => {
      alert("Error 325 ")
    })
  }
(…)
```

< React.js (frontend) />

- Installer o npm install react-validation
- Na pasta '**views**', crie o ficheiro '**login.js**', com o seguinte código:

```
import React, { useEffect, useState } from "react";
import Form from "react-validation/build/form";
import Input from "react-validation/build/input";
import CheckButton from "react-validation/build/button";
import AuthService from "../view/auth.service";
import { useNavigate } from "react-router-dom";

const required = (value) => {
  if (!value) {
    return (
      <div className="alert alert-danger" role="alert">
        Este campo é de preenchimento obrigatório!
      </div>
    );
  }
};
```

< React.js (frontend) - Login />

```
export default function LoginComponent() {
  const [username, setusername] = useState("");
  const [password, setpassword] = useState("");
  const [loading, setloading] = useState(false);
  const [message, setmessage] = useState("");
  const navigate = useNavigate();

async function HandleLogin(event) {
    event.preventDefault();
    setmessage("");
    setloading(true);

      AuthService.login(username, password)
        .then((res) => {
          if (res === "" || res === false) {
            setmessage("Autenticação falhou.");
            setloading(false);
          }
          else {
              navigate("/");
            }
          })
          .catch((error) => {
            setmessage("Autenticação falhou.");
            setloading(false);
          });

  }
```

< React.js (frontend) - Login />

```
return (
    <div className="col-md-4">
      <div className="card card-container">
        <img
          src="//ssl.gstatic.com/accounts/ui/avatar_2x.png"
          alt="profile-img"
          className="profile-img-card"
        />

        <Form onSubmit={HandleLogin}>
          <div className="form-group">
            <label htmlFor="username">Username</label>
            <Input
              type="text"
              className="form-control"
              name="username"
              value={username}
              onChange={(value) => setusername(value.target.value)}
            />
          </div>
```

< React.js (frontend) - Login />

```
<div className="form-group">
  <label htmlFor="password">Password</label>
  <Input
    type="password"
    className="form-control"
    name="password"
    value={password}
    onChange={(value) => setpassword(value.target.value)}
  />
</div>

<div className="form-group">
  <button className="btn btn-primary btn-block">
    <span>Login</span>
  </button>
</div>
```

< React.js (frontend) - Login />

```
{message && (
            <div className="form-group">
              <div className="alert alert-danger" role="alert">
                {message}
              </div>
            </div>
          )}
        </Form>
      </div>
    </div>
  );
}
```

< React.js (frontend) />

- Por último é necessário realizar algumas alterações no ficheiro principal '**app.js**';
  - Importação do sistema de autenticação e dos componentes criados anteriormente:

```
(…)
import AuthService from "./view/auth.service";
import Login from "./view/login";
(…)


export default function AppComponent() {
  const [currentUser, setcurrentUser] = useState("");

  useEffect(() => {
    const user = AuthService.getCurrentUser();
    if (user) {
      setcurrentUser({ currentUser: user });
    }
  }, []);
```

< React.js (frontend) />

- Definição da função de logout():

```
logOut() {
  AuthService.logout();
}
```

< React.js (frontend) />

- A definição das novas rotas de autenticação no React:

```
<Routes>
   (…)
       <Route path="/login" element={<Login />} />
</Routes>
```