JSON Web Token (JWT) is an open standard ([RFC 7519](https://tools.ietf.org/html/rfc7519)) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed. JWTs can be signed using a secret (with the **HMAC** algorithm) or a public/private key pair using **RSA** or **ECDSA**

<https://jwt.io/introduction/>

Back

|  |  |
| --- | --- |
| Створюємо модель для користувача | const mongoose = require(‘mongoose’);  const crypto = require(‘crypto’);  const { Schema } = mongoose;  const UsersSchema = new Schema({  email: String,  nick: String,  hash: String, //поле, де буде зберігатися хеш пароля  salt: String, //поле, де буде зберігатися ключ  });  //--------------- Функція для формування хешу пароля -----------------  UsersSchema.methods.setPassword = function(password) {  this.salt = crypto.randomBytes(16).toString(‘hex’);  this.hash = crypto.pbkdf2Sync(password, this.salt, 10000, 512, ‘sha512’).toString(‘hex’);  };  //---------------- Функція для перевірки правильності пароля ------------  UsersSchema.methods.validPassword = function(password) {  //----------- Формуємо хеш переданого (для перевірки) пароля ----  const hash = crypto.pbkdf2Sync(password, this.salt, 10000, 512, ‘sha512’).toString(‘hex’);  //------------ Перевіряємо, чи одержано такий же хеш як у базі -------------  return this.hash === hash;  };  module.exports= mongoose.model(‘Users’, UsersSchema); |
| Створюємо роут для роботи з користувачами | var express = require(‘express’);  var router = express.Router();  const bodyParser = require(“body-parser”);  const urlencodedParser = bodyParser.urlencoded({extended: false});  const User = require(‘../app\_api/models/user’)  const { prepareToken } = require(‘../utils/token’);  router.post(‘/signup’, urlencodedParser,  function (req,res) {  var user=new User({  email:req.body.email,  nick:req.body.nick,  });  user.setPassword(req.body.password);  user  .save()  .then((user) => {  const token = prepareToken(  {  id: user.\_id,  nick: user.nick  },  req.headers  );  return res.status(201).json({  result: ‘Signuped successfully’,  token  });  })  .catch((err) => {  return res.status(500).json({ error: ‘Signup error’ });  });  }  )  router.post(‘/login’, urlencodedParser,  function (req,res) {    if (!req.body.email) {  return res.status(401).json({ error: ‘Email is required’ });  }  if (!req.body.password) {  return res.status(401).json({ error: ‘Password is required’ });  }    User.findOne({ email: req.body.email })  .exec()  .then((user) => {    if (!user) {  return res.status(401).json({ error: ‘User not found’ });  }  if (!user.validPassword(req.body.password)) {  return res.status(401).json({ error: ‘Login error’ });  }  const token = prepareToken(  {  id: user.\_id,  nick: user.nick  },  req.headers  );  res.json({  result: ‘Authorized’,  token  });  })  .catch((err) => {  return res.status(401).json({ error: ‘Login error’ });  });  }  )  module.exports = router; |
| Модуль для роботи з JWT  npm install jsonwebtoken  <https://www.npmjs.com/package/jsonwebtoken> | const jwt = require('jsonwebtoken');  const config =require('../config')  function parseBearer(bearer, headers) {  if (bearer.startsWith('Bearer ')) {  token = bearer.slice(7, bearer.length);  }  const decoded = jwt.verify(token, prepareSecret(headers));  return decoded;  }  function prepareToken(data, headers) {  return jwt.sign(data, prepareSecret(headers), { expiresIn: config.tokenExpires });  }  function prepareSecret(headers) {  return config.tokenKey + headers['user-agent'] + headers['accept-language'];  }  module.exports = { parseBearer, prepareToken }; |
| Додаємо middleware для перевірки авторизації користувача | const {parseBearer} =require('./utils/token')  app.use(function(req, res, next) {  res.header('Access-Control-Allow-Origin', '\*');  res.header('Access-Control-Allow-Headers', 'Origin, X-Requested-With, Content-Type, Accept, Authorization');  res.header('Access-Control-Allow-Methods', 'GET, POST, OPTIONS, PUT, PATCH, DELETE');  next();  });  app.use((req, res, next) => {  const openPathes = [ '/users/login', '/users/signup', '/api/products' ];  if (!openPathes.includes(req.path)) {  try {  req.user = parseBearer(req.headers.authorization, req.headers);  } catch (err) {  return res.status(401).json({ result: 'Access Denied' });  }  }  next();  });  //- |
|  |  |
|  |  |
|  |  |
|  |  |

Клієнт

|  |  |
| --- | --- |
| Додаємо до кожного запиту axios передачу token  (наприклад в index.js) | axios.interceptors.request.use(function(config) {  if (localStorage.token) {  config.headers['Authorization'] = 'Bearer ' + localStorage.token;  } else {  config.headers['Authorization'] = null;  }  return config;  }); |
|  |  |
| Створюємо функцію, що повертає компонент вищого порядку для контролю захищених роутів | import React, { Component } from 'react';  import {withRouter} from 'react-router-dom'  export default function authRequired(ProtectedComponent){  class Authorized extends Component {  componentWillMount(){  if(!localStorage.token)  this.props.history.push('/login')  }  componentWillUpdate(){  if(!localStorage.token)  this.props.history.push('/login')  }  render() {  return (  <ProtectedComponent {...this.props}/>  );  }  }    return withRouter( Authorized)  } |
| Захищаємо компоненти з використанням компоненту вищого порядку  authRequired(**компонент**) | import React, { Component } from 'react';  import ProductsViewer from '../products-viewer'  import AddProduct from '../add-product'  import {Link, Route,Switch, Redirect} from 'react-router-dom';  import EditProduct from '../edit-product/edit-product';  import Home from '../home'  import authRequired from '../../utils/authRequired'  import Login from '../login'  import Signup from '../signup'  class Main extends Component {  constructor(props) {  super(props);  }  logout=()=>{  this.props.unsetUser()  }  render() {  const {isAuth}=this.props  return (  <>  <div>  <ul>  <li><Link to="/">Home</Link></li>  <li><Link to="/products">Products</Link></li>  <li><Link to="/products/add">Add</Link></li>  </ul>  {  !isAuth?(  <>  <button>  <Link to="/login">Login</Link>  </button>  <button>  <Link to="/signup">Signup</Link>  </button>  </>  ):  <button onClick={this.logout}>Logout</button>  }  </div>  <hr/>  <div>  <Switch>  <Route path="/" exact component={Home} />  <Route path="/products" exact component={ProductsViewer} />  <Route path="/products/add" exact component={authRequired(AddProduct)} />  <Route path="/products/edit/:productId" exact component={authRequired(EditProduct)} />  <Route path="/login" exact component={Login} />  <Route path="/signup" exact component={Signup} />  <Route component={Home} />  </Switch>;  </div>  </>  );  }  }    export default Main; |
| Елементи Store для роботи з авторизацією | -----------------------------  const ActionTypes={  SET\_USER:'SET\_USER',  UNSET\_USER:'UNSET\_USER'  }  export default ActionTypes  -----------------------------  import ActionTypes from './action-types'  export function setUser(token) {  return {  type:ActionTypes.SET\_USER,  payload:token  }  }  export function unsetUser() {  return {  type:ActionTypes.UNSET\_USER  }  }  -------------------------------  import ActionTypes from './action-types'  const usersReducer=(state,action)=>{  switch (action.type) {  case ActionTypes.SET\_USER:  localStorage.setItem('token',action.payload)  return {  isAuth:true  }  case ActionTypes.UNSET\_USER:  localStorage.removeItem('token')  return {  isAuth:false  }  default:  return state;  }  }  export default usersReducer  -------------------------------  import { createStore } from "redux";  import reducers from "./reducers";  let initialStore= {  isAuth:false  } ;  export default createStore(reducers,initialStore);  -------------------------------  -------------------------------  -------------------------------  ------------------------------- |
| Сторінка авторизації | import React, { Component } from 'react';  import {withRouter} from 'react-router-dom'  import axios from 'axios'  class Login extends Component {  constructor(props) {  super(props);  this.state={  errorMsg:''  }  }  sendData=async (e)=>{  e.preventDefault()  const user={  'email':this.refs.email.value,  'password':this.refs.password.value,  }    try {  const response = await axios.post('http://localhost:3000/users/login', user);  this.props.setUser(response.data.token)  this.props.history.push('/')  return true;  } catch (err) {  throw err;  }  }  render() {  const {errorMsg}= this.state  return (  <>  <form >  <label>  Email :  <input ref="email" type='email' />  </label>  <label>  Password :  <input ref="password" type='password' />  </label>  <input type='submit' value='send' onClick={(e)=>this.sendData(e)}/>  </form>  {  errorMsg?errorMsg:''  }  </>  );  }  }  export default withRouter(Login); |
|  |  |
|  |  |