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 роутер для роботи з користувачами | var express = require('express');  var router = express.Router();  const User = require('../models/user')  const { prepareToken } = require('../utils/token');  router.post('/signup',    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',    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                );                const expiresAt=new Date().getTime() +36000000                res.status(200).json({                    result: 'Authorized',                    user:{                        authData:{                        nick:user.\_doc.nick,                        access\_token:token                      },                      expiresAt                    }                });            })        .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> | //---------------------- /utils/token.js ------------------------------  const jwt = require('jsonwebtoken');  const expiresIn = '60m'  const tokenKey = 'Our Token Key'   // Save in .env !!!  function parseBearer(bearer, headers) {      let token=null      if (bearer.startsWith('Bearer ')) {         token = bearer.slice(7, bearer.length);         console.log('token');         console.log(token);        }      const decoded = jwt.verify(token, prepareSecret(headers));      return decoded;  }  function prepareToken(data, headers) {      return jwt.sign(data, prepareSecret(headers), { expiresIn: '60m'});  }  function prepareSecret(headers) {      return tokenKey + headers['user-agent'] + headers['accept-language'];  }  module.exports = { parseBearer, prepareToken }; |
| Додаємо middleware для перевірки авторизації користувача (app) | 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', '/products' ];    const openPathes = [ '/users/login', '/users/signup' ];    if (!openPathes.includes(req.path)) {        try {          console.log('req.headers.authorization');          console.log(req.headers.authorization);              req.user = parseBearer(req.headers.authorization, req.headers);        } catch (err) {            return res.status(401).json({ result: 'Access Denied' });        }    }    next();  }); |
|  |  |
|  |  |
|  |  |
|  |  |

**------------------- Front ----------------------**

|  |  |
| --- | --- |
| auth - Модуль vuex для роботи з авторизацією | import axios from 'axios'  import apiEndpoints from "@/constants/apiEndpoints";  export default {      namespaced: true,      state: {        authData: JSON.parse(localStorage.getItem('authData')) || null,        expiresAt: localStorage.getItem('expiresAt') || null,      },      mutations: {        setAuthData(state, { authData, expiresAt }) {          state.authData = { ...authData }          state.expiresAt = expiresAt || state.authData.expires\_in \* 1000 + new Date().getTime()            localStorage.setItem('authData', JSON.stringify(state.authData))          localStorage.setItem('expiresAt', JSON.stringify(state.expiresAt))        },        clearAuthData(state) {          state.authData = null          state.expiresAt = null            localStorage.removeItem('authData')          localStorage.removeItem('expiresAt')        }      },      actions: {        setAuthData({ commit }, { authData, expiresAt }) {            commit('setAuthData', { authData, expiresAt })        },        signup({ commit }, { nick, email, password }) {          return new Promise((resolve, reject) => {              axios.post(apiEndpoints.user.signup, { nick, email, password })              .then(function() {              //   commit('setAuthData', { authData: user.data })                resolve(true)              })              .catch(err => {                commit('clearAuthData')                reject(err)              })          })        },        login({ commit }, { email, password }) {          return new Promise((resolve, reject) => {                axios.post(apiEndpoints.user.login, { email, password })              .then(res=>res.data)              .then(data=> {                    commit('setAuthData', { ...data.user})                resolve(true)              })              .catch(err => {                commit('clearAuthData')                reject(err)              })          })        },          logout({ commit }) {          commit('clearAuthData')        }      },      getters: {        isAuthenticated: state => () => {            return state.authData && new Date().getTime() < state.expiresAt        },        getAccessToken: state=>()=> {          return state.authData && state.authData.access\_token        },        authorized:state=>state.authData && new Date().getTime() < state.expiresAt      }    } |
|  | //------------------------------ main.js ---------------------------------  import Vue from "vue";  import App from "./App.vue";  import store from "./store";  import axios from "axios";  //Імпортуємо  Buefy  import Buefy from "buefy";  import "buefy/dist/buefy.css";  Vue.use(Buefy);  //Імпортуємо router  import router from "./router  Vue.config.productionTip = false; |
| Додаємо до кожного запиту axios на сервер передачу token  (у загловок додаємо додаткове поле 'Authorization') | axios.interceptors.request.use(function(config) {    const isAuthenticated = store.getters["auth/isAuthenticated"]();    if (isAuthenticated) {      config.headers["Authorization"] =        "Bearer " + store.getters["auth/getAccessToken"]();    } else {      config.headers["Authorization"] = null;    }    return config;  }); |
| При надходженні відповіді від сервера перевіряємо чи користувач авторизований | axios.interceptors.response.use(    function(response) {      return response;    },    function(error) {      if (error.response.status === 401) {      //🡨 Якщо сервер відповів «401»        router.push({ path: "/login" });      }      return Promise.reject(error);    }  ); |
|  | router.beforeEach((to, from, next) => {    let check =      !store.getters["auth/isAuthenticated"]() &&      !["/login", "/signup"].includes(to.path);    if (check) {    //🡨 Недопускаємо до захищених роутів, якщо немає токена      next({ path: "/login" });      return;    } else {      next();    }  }); |
|  | new Vue({      store,      router,      render: (h) => h(App)  }).$mount('#app'); |
| Login page | <template>      <div>          <h2>Login</h2>          <form @submit.prevent="submit">              <div> Please log in </div>              <div >                  email :                  <input                      type="text"                      v-model="email"                      name="email"                      placeholder="Email"                  >              </div>              <div class="form-group">                  password :                  <input                      type="password"                      v-model="password"                      name="password"                      placeholder="Password"                  >              </div>              <div class="form-group error" v-if="message">{{ message }}</div>              <div class="form-group">                  <button class="btn btn-primary">Login</button>                  <router-link to="/signup" class="btn btn-link">Sign up</router-link>              </div>          </form>      </div>  </template>  <script>  import { mapActions } from 'vuex';  export default {      name: "Login",      data() {          return {              email: "",              password: "",              message: ""          };      },      methods: {          ...mapActions('auth',['login', 'logout']),          async submit() {              try {                  const user = { email: this.email, password: this.password };                  const result = await this.login(user);                    if (result === true) {                      this.message = "";                        this.$router.push({ path: "/" });                  } else {                      this.message = "Login error!";                  }              } catch (err) {                  this.message = err.response.data.error;              }          }      },      created() {          this.logout();      }  };  </script> |
| Signup page | <template>      <div>          <h2>Sign up</h2>          <form @submit.prevent="submit">              <div >                  Nick :                  <input                      type="text"                      v-model="nick"                      name="nick"                      placeholder="Nick"                  >              </div>              <div>                 email :                  <input                      type="text"                      v-model="email"                      name="email"                      placeholder="Email"                  >              </div>              <div class="form-group">                  Password : <input                      type="password"                      v-model="password"                      name="password"                      placeholder="Password"                  >              </div>              <div  v-if="message">{{ message }}</div>              <div >                  <button type="submit" >Sign up</button>              </div>          </form>      </div>  </template>  <script>  import { mapActions } from "vuex";  export default {      name: "Signup",      data() {          return {              nick: "",              email: "",              password: "",              message: ""          };      },      methods: {          ...mapActions('auth',['signup', 'logout']),          async submit() {                try {                  const user = {                      nick: this.nick,                      email: this.email,                      password: this.password                  };                  const result = await this.signup(user);                  if (result === true) {                      this.message = "";                      this.$router.push({                          path: "/login",                          query: { signedup: "true" }                      });                  } else {                      this.message = result; //'SignUp error!';                  }              } catch (err) {                  this.message = err.message;              }          }      },      created() {          this.logout();      },  };  </script> |
| App.vue | <template>    <div id="app">      <!-- Навігаційна панель -->      <b-navbar>        <template slot="brand">          <b-navbar-item tag="router-link" :to="{ path: '/' }">            <img              src="https://raw.githubusercontent.com/buefy/buefy/dev/static/img/buefy-logo.png"              alt="Lightweight UI components for Vue.js based on Bulma"            />          </b-navbar-item>        </template>        <template slot="start">          <b-navbar-item tag="router-link" :to="{ path: '/' }"            >Home</b-navbar-item          >          <b-navbar-item tag="router-link" :to="{ path: '/create' }"            >Add product</b-navbar-item          >          <b-navbar-item tag="router-link" :to="{ path: '/edit' }"            >Edit product</b-navbar-item          >        </template>        <template slot="end">          <b-navbar-item tag="div">            <b-icon pack="fas" icon="shopping-cart" size="is-small"></b-icon>            <b-tag type="is-info is-rounded">{{ getProductsCountInCart }}</b-tag>            <div class="buttons">                <router-link v-if="!isAuthenticated()" to="/signup">                <strong>Sign up</strong>              </router-link>              <router-link v-if="!isAuthenticated()" to="/login">                <strong>Log in</strong>              </router-link>              <button v-if="isAuthenticated()" @click="onLogout">Log out</button>            </div>          </b-navbar-item>        </template>      </b-navbar>      <!-- Місце вставки сторінок -->      <router-view></router-view>    </div>  </template>  <script>  import { mapGetters, mapActions } from "vuex";  export default {    name: "App",    computed: {      ...mapGetters("cart", ["getProductsCountInCart"]),      ...mapGetters("auth", ["isAuthenticated"]),      cartLenght() {        return this.getProductsCountInCart();      },    },    methods: {      ...mapActions('auth',['logout']),      onLogout() {        this.logout()         this.$router.push({ path: "/login" });      }    },  };  </script>  <style>  #app {    font-family: Avenir, Helvetica, Arial, sans-serif;    -webkit-font-smoothing: antialiased;    -moz-osx-font-smoothing: grayscale;    text-align: center;    color: #2c3e50;    margin-top: 60px;  }  </style> |
|  |  |
|  |  |
|  |  |
|  |  |