Użytkownicy

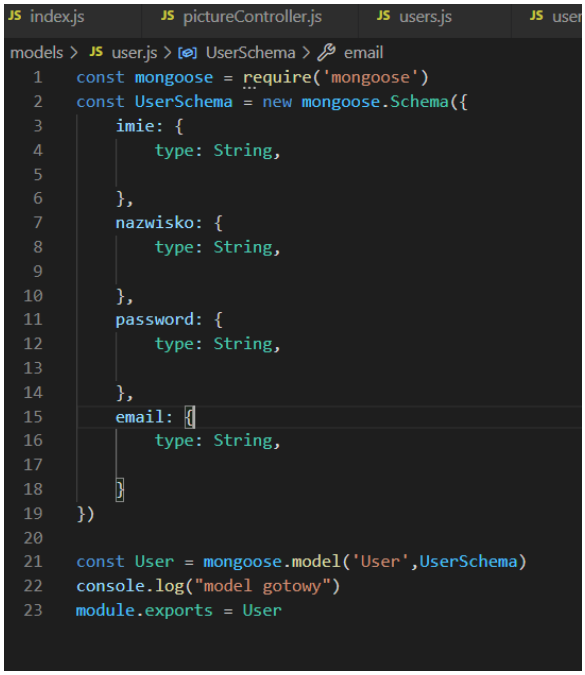
W tym tutorialu przedstawiony będzie system logowania i rejestracji oraz autentyfikacja. W przypadku wielu serwisów niezbędna funkcja. Aby stworzyć ten system na początku musimy zmodyfikować ścieżki i dodać kontroler oraz model dla użytkowników. Do tego kursu będą potrzebne dwa moduły:

npm install jsonwebtoken

npm install bcryptjs

Dobrym zwyczajem, a nawet obowiązkiem jest zabezpieczenie również hasła w bazie danych, czyli np. haszowanie. Dlatego to będzie pierwszym celem przy rejestracji

Na początku ustalmy model w folderze models/user.js:

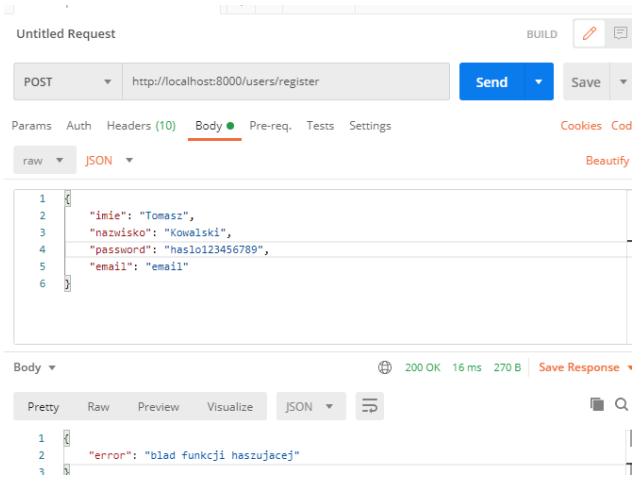


W tym przypadku jest to podstawowa wersja modelu, oczywiście warto dodać zabezpieczenia typu pola wymagane, ograniczenia np. co do długości hasła. Następnie możemy utworzyć ścieżki i na początku niech będzie funkcja do rejestrowania użytkowników - register.

W pliku router/users.js należy dodać wymaganą paczkę Express, obiekt router oraz zmienną która będzie przetrzymywała ścieżkę do kontrolera. Rejestracja będzie odbywała się za pomocą funkcji register. Następnie należy stworzyć kontroler i uzupełnić go o funkcję tworzącą użytkownika.

Controllers/userController.js

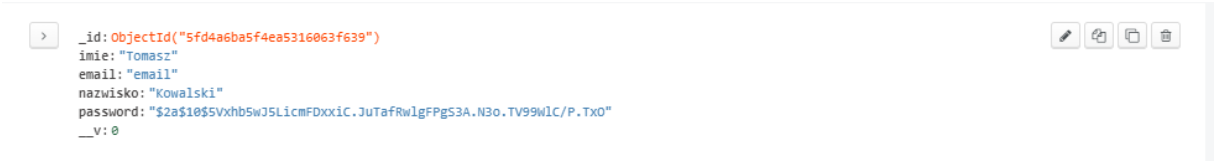
Na początku należy dołączyć odpowiednie moduły oraz ścieżkę do modelu. Funkcja register na początku próbuje wykonać funkcję obiektu bcrypt która będzie haszowała zawartość. Pierwszy argument to string który chcemy zahaszować, drugi to liczba salt round odpowiedzialna za poziom zahaszowania oraz callback. Wewnątrz sprawdzamy czy nie wystąpił jakiś błąd, jeżeli nie to kolejno tworzony jest użytkownik na podstawie parametrów z req.body a następnie zapisywany do bazy. Komunikat zwrotny zostanie wyświetlony w formacie json. Teraz możemy przetestować rejestrację za pomocą Postman.



Niestety otrzymaliśmy błąd. Po sprawdzeniu zawartości hasła okazuje się że nie zostało przechwycone. Aby to naprawić wystarczy dodać linijkę w pliku głównym index.js odpowiedzialną za ustawienie bodyParsera na posługiwanie się formatem json:

```
22    app.use(bodyParser.json());
23
24    app.use('/users', usersRoutes)
25    app.use('/pictures', picturesRoutes)
26
```

Po ponownym uruchomieniu Postmana i wysłaniu danych użytkownika powinniśmy otrzymać komunikat że dodano użytkownika oraz w mongoDBCompass możemy zobaczyć że dostał dodany:

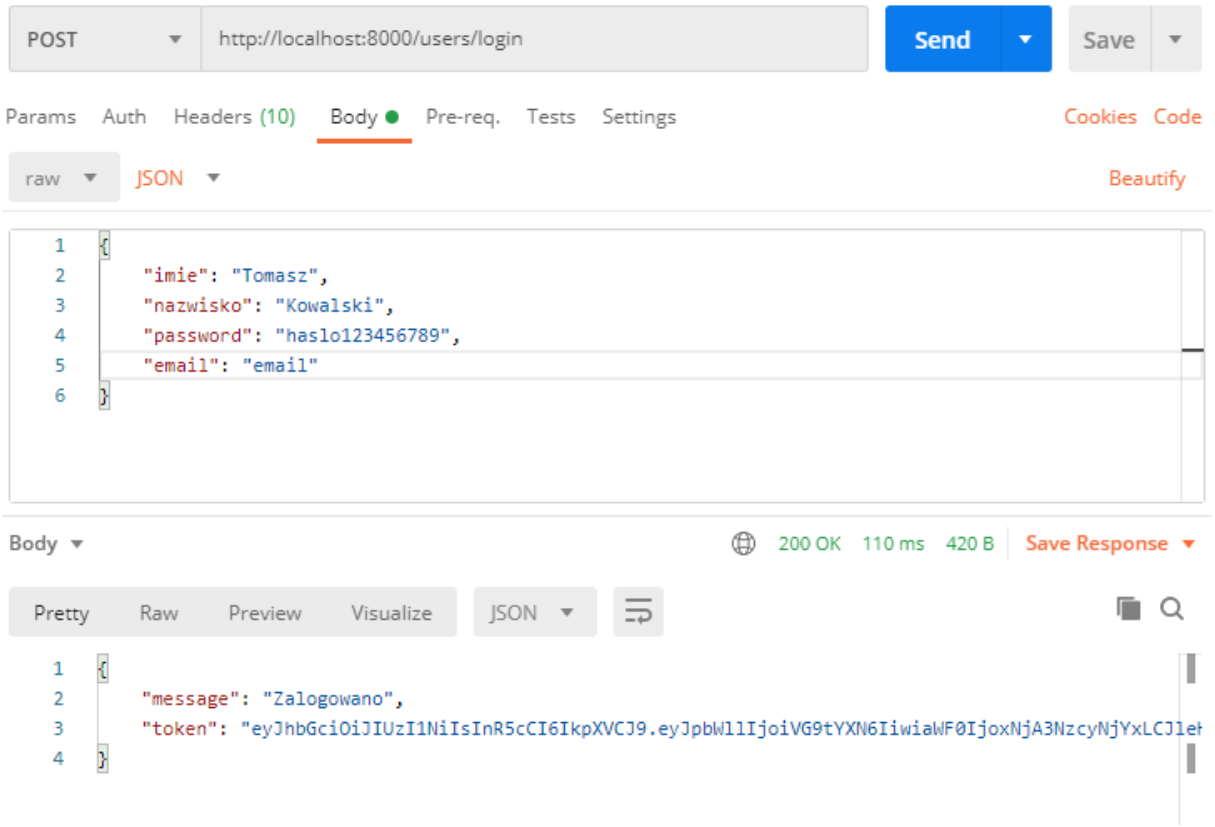


Pole password jest zahaszowane, zatem wszystko się udało. Teraz można przejść do logowania. Funkcja logowania powinna sprawdzić czy hasz zgadza się z podanym hasłem a następnie wygenerować token i dać odpowiedni komunikat.

controllers > JS userController.js > login > exports.login > then() callback

```
35  exports.login = (req, res, next) => {
36      var nazwisko = req.body.nazwisko
37      var password = req.body.password
38
39      User.findOne({nazwisko})  // login jako nazwisko
40          .then(user => {
41              if (user) {
42                  //poruwnujemy hasło podane w logowaniu z hasłem użytkownika
43                  bcrypt.compare(password, user.password, function (err, result) {
44                      if (err) {
45                          res.json({
46                              error: err
47                          })
48                      }
49                      if (result) {
50                          //tworzymy token, gdzie 'kodSzyfrujacy' to tajny kod na podstawie którego
51                          //generowany jest hash, będzie on potrzebny do sprawdzenia tokena później
52                          let token = jwt.sign({ imie: user.imie }, 'kodSzyfrujacy', { expiresIn: '1h' })
53                          res.json({
54                              message: 'Zalogowano',
55                              token: token
56                          })
57                      } else {
58                          res.json({
59                              message: 'Zle haslo'
60                          })
61                      }
62                  })
63              } else {
64                  res.json({
65                      message: 'No user found!'
66                  })
67              }
68          })
69  }
```

W Postmanie można teraz sprawdzić czy uda się zalogować.



Użytkownik został zalogowany i dostał token który będzie służył za dostęp do zabezpieczonych stron.

Autentyfikacja

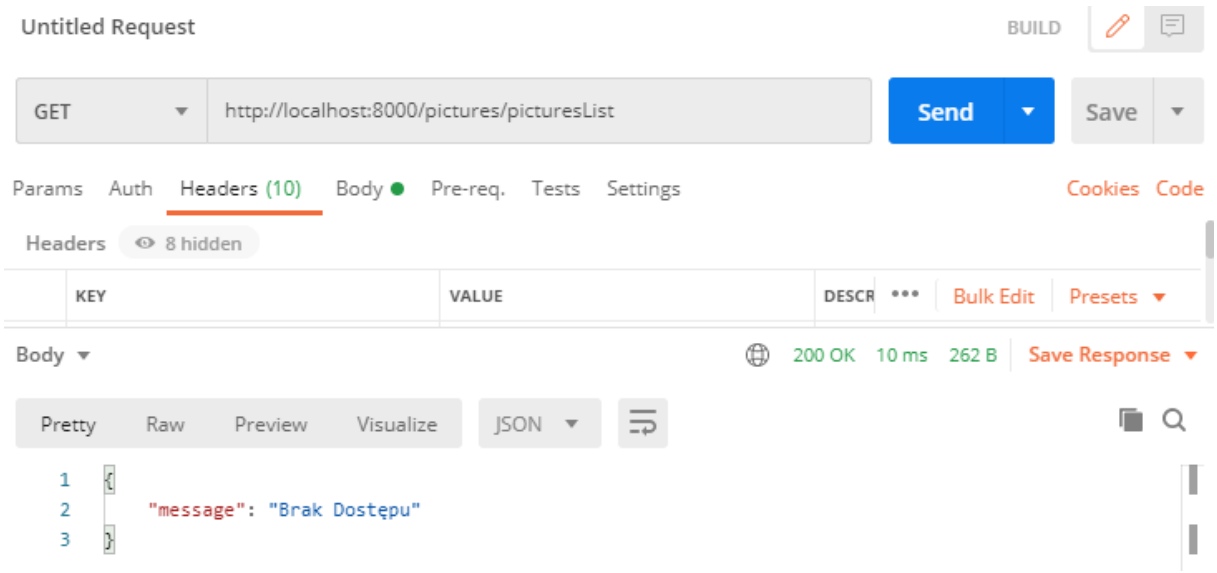
Aby zabezpieczyć serwis przed osobami niezarejestrowanymi lub nie mającymi praw dostępu można skorzystać z systemu autentyfikacji. W nodejs można to zrobić za pomocą middleware. System ten opiera się o dodatkowe funkcje pomiędzy żądaniem a wywołaniem funkcji, dzięki temu użytkownik niespełniający wymagań danego middleware nie uzyska dostępu do danej podstrony. Będzie to spełnione za pomocą tokena, czyli wygenerowanego ciągu znaków dla konkretnej sesji użytkownika. Do stworzenia takiego systemu potrzebujemy folder middleware w folderze głównym, modułu npm jsonwebtoken (https://www.npmjs.com/package/jsonwebtoken) odpowiedzialny za tworzenie i porównywanie tokenów.

Na początku należy stworzyć kod odpowiedzialny za sprawdzenie tokena czy jest prawidłowy. Będzie to funkcja middleware dlatego najlepiej zamieścić kod w pliku middleware/authenticate.js. Funkcja na początku powinna pobrać z nagłówka token, następnie funkcją jwt.verify sprawdzić czy zgadza się z ustawionym wcześniej kodem szyfrującym a następnie zadecydować o dostępie.

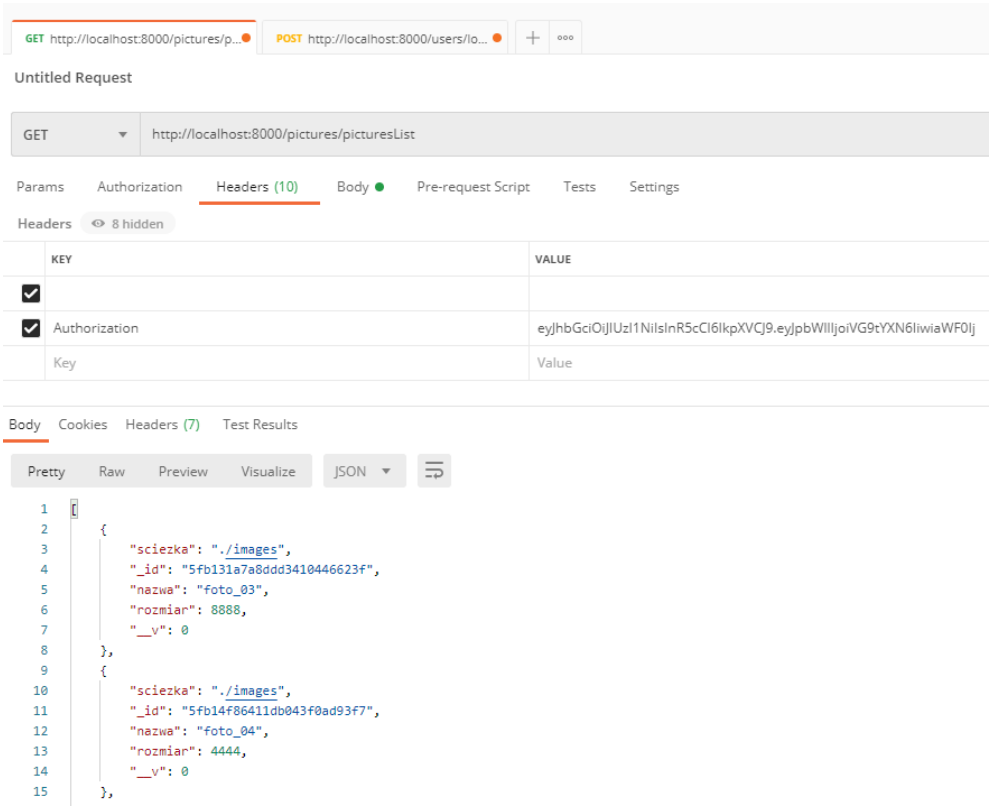
```
middleware > JS authenticate.js > ...
1   const jwt = require('jsonwebtoken')
2
3   const authenticate = (req, res, next) => {
4       try {
5
6           const token = req.headers.authorization
7           const decode = jwt.verify(token, 'kodSzyfrujacy')
8
9           req.user = decode
10          next()
11      }
12      catch (err) {
13          res.json({
14              message: 'Brak Dostępu'
15
16          })
17      }
18  }
19
20  module.exports = authenticate
```

Teraz możemy zablokować jedną z podstron, np. wylistowanie zawartości bazy danych zdjęć. Aby to zrobić wystarczy w routes/pictures.js dodać utworzony middleware.

```
routes > JS pictures.js > ...
1   const express = require('express')
2   const router = express.Router();
3   var picture_controller = require('../controllers/pictureController');
4
5   //odniesienie do middleware
6   var authenticate = require('../middleware/authenticate');
7
8   //aby skorzystać z autentyfikacji w danej podstronie wystarczyy dodać
9   router.get('/picturesList',authenticate, picture_controller.picture_list);
10  //Dostęp do funkcji pod adresem http://localhost:8000/pictures/picturesList
11
```



Jak widać nie mamy dostępu do strony ponieważ nie został przesłany token zalogowanego użytkownika. Żeby przetestować możemy w nagłówku wysłać token zalogowanego użytkownika. Aby to zrobić należy zalogować się przez Postman a następnie skopiować token i wkleić w headers: Authorization. To pole jest sprawdzane w middleware odpowiedzialne za dostęp.



Tym razem udało się dostać do wylistowania bazy zdjęć. Ostatnim krokiem może być zapisanie tokenu w cookies albo lokalnie. Można skorzystać z modułu:

https://www.npmjs.com/package/node-localstorage

Instalacja:

npm install node-localstorage

Przykład użycia:

```
require("node-localstorage")

let token = "12345"

if (typeof localStorage === "undefined" || localStorage === null) {
    var LocalStorage = require('node-localstorage').LocalStorage;
    localStorage = new LocalStorage('./scratch');
}

localStorage.setItem('token', token);
console.log(localStorage.getItem('token'));
```

Zadania

Skonstruuj system logowania i rejestracji oparty na formularzach. Zablokuj możliwość edycji i modyfikacji zdjęć użytkownikom niezalogowanym.