MEAN Stack - obsługa poczty elektornicznej, biblioteka Nodemailer

Tworzenie serwisów Web 2.0

dr inż. Robert Perliński
rperlinski@icis.pcz.pl

22 marca 2019

Plan prezentacji

Spis treści

1 Obsługa poczty elektronicznej

Biblioteki obsługujące pocztę elektroniczną w Node.js

Biblioteki obsługujące pocztę elektroniczną w Node.js:

- **Node**mailer - https://nodemailer.com - najlpesza, najbardziej rozbudowana, jedna z nielicznych, która ma własną dokumentację, inną niż na npmjs.com czy github.com,
- express-mailer - https://www.npmjs.com/package/express-mailer
- express-mail - https://www.npmjs.com/package/express-mail
- mail - https://www.npmjs.com/package/mail
- mail2 - https://www.npmjs.com/package/mail2

Przestarzałe, dołączone do biblioteki Nodemailer:

- mailcomposer - https://nodemailer.com/extras/mailcomposer/, nie pozwala na wysyłanie maili tylko na ich komponowanie i wysyłanie do otwartego już kanału SMTP czy do pliku
- mailparser - https://nodemailer.com/extras/mailparser/, do parsowania maili, nawet bardzo dużych i w dużych ilościach...

1.1 Nodemailer

Biblioteka Nodemailer

https://nodemailer.com

O bibliotece Nodemailer:

- proste wysyłanie maili (ang. easy as cake - proste jak ...),
- projekt rozpoczął się przed rokiem 2010 kiedy nie było jakiejś sensownej biblioteki to wysyłania poczty,
- dzisiaj Nodemailer jest niemal domyślnie wybierany do obsługi poczty elektronicznej w Node.js,
- obecna wersja ma numer 5.1.1 i jest na licencji MIT,
- istnieje też płatna licencja komercyjna.

Biblioteka Nodemailer I

Możliwości biblioteki Nodemailer:

- pojedynczy moduł bez żadnych dodatkowych zależności - kod łatwy w zarządzaniu,
- duży nacisk położono na bezpieczeństwo, brak takich usterek jak RCE (Remote Code Execution),
- wsparcie dla Unicode pozwala na użycie dowolnych znaków, również emoji,
- wsparcie dla Windows, wystarczy zainstalować jak dowolny inny moduł,
- treść wiadomości może być tekstowa albo w HTML,
- pozwala na dodawanie załączników,
- możliwość umieszczania obrazów z załącznika w treści HTML wiadomości,

Biblioteka Nodemailer II

Możliwości biblioteki Nodemailer:

- bezpiecznie przesyłanie wiadomości dzięki TSL/STARTTLS,
- różne metody przesyłania danych poza domyślnym protokołem SMTP,
- możliwość podpisywania wiadomości kluczem DKIM (DomainKeys Identified Mail),
- system wtyczek do modyfikowania wiadomości,
- dodatkowy system autoryzacji OAuth2 korzystający z tokenów,
- obsługa Proxy dla połączeń SMTP,
- obsługa kodu ES6 - brak niezamierzonych wycieków pamięci dzięki obsłudze `var`.

Instalacja i wymagania:

- instalacja: `npm install nodemailer --save`,
- Nodemailer wymaga tylko Node.js w wersji 6+, żadnych innych specyficznych wymagań.

1.2 gmail

Użycie Nodemailer w skrócie - TL;DR

1. Tworzymy warstwę transportową używając SMTP (domyślnie) albo jakąś inną.
2. Ustawiamy wszystkie dane odnośnie wiadomości (kto, do kogo, treść, załączniki,...).
3. Dostarczamy wiadomość używając metody `sendMail()` utworzonej wcześniej warstwy transportowej.

Użycie Nodemailer w skrócie - TL;DR

1. Tworzymy warstwę transportową używając SMTP (domyślnie) albo jakąś inną.

utils/mailerTransporter.js

```
const nodemailer = require('nodemailer');

// tworzy obiekt warstwy transportowej, domyślnie używa protokołu SMTP
let gmailTransporter = nodemailer.createTransport({
    service: 'gmail',
    auth: {
        user: 'rperlinski.pcz@gmail.com',
        pass: 'SuperMegaTajneHaslo'
    }
});

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

Użycie Nodemailer w skrócie - TL;DR

2. Ustawiamy wszystkie dane odnośnie wiadomości (kto, do kogo, treść, załączniki,...).

index.js

```
router.get('/send-email', function(req, res, next) {
  // wysyła wiadomości, dane mogą byc w unicode
  let mailOptions = {
    // adres nadawcy
    from: '"Robert Perliński" <rperlinski.pcz@gmail.com>',
    // lista odbiorców
    to: 'pierwszy.adres@gmail.com, rperlinski@icis.pcz.pl',
    // temat wiadomości
    subject: 'Testowa wiadomość',
    // treść wiadomości tekstowej
    text: 'Treść wiadomości jako czysty tekst',
    // treść wiadomości w html
    html: `<b>Treść wiadomości w HTML</b>
    <p>Jakieś znaki w unicode, grecki alfabet: α, β, γ, ...</p>`
  };

  ...
});
```

Użycie Nodemailer w skrócie - TL;DR

3. Dostarczamy wiadomość używając metody `sendMail()` utworzonej wcześniej warstwy transportowej.

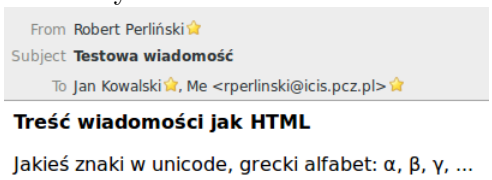
index.js

```
router.get('/send-email', function(req, res, next) {
  ...

  // wysyła maila dla ustawionej warstwy transportowej dla danych opcji
  gTransporter.sendMail(mailOptions, (error, info) => {
      if (error) {
          return console.log(error);
      }
      console.log('Wiadomość %s wysłana: %s',
                  info.messageId, info.response);
      res.send('Wiadomość ' + info.messageId +
               ' wysłana: ' + info.response);
  });
});
```

```
Wiadomość <51c06368-d687-ef1a-5785-c581e9db0a36@gmail.com>
wysłana: 250 2.0.0 OK 1490142828 a16sm3936955lfk.24 - gsmtp
```

Odebrany mail

Odebrany mail w formie HTML:



Odebrany mail w formie tekstowej

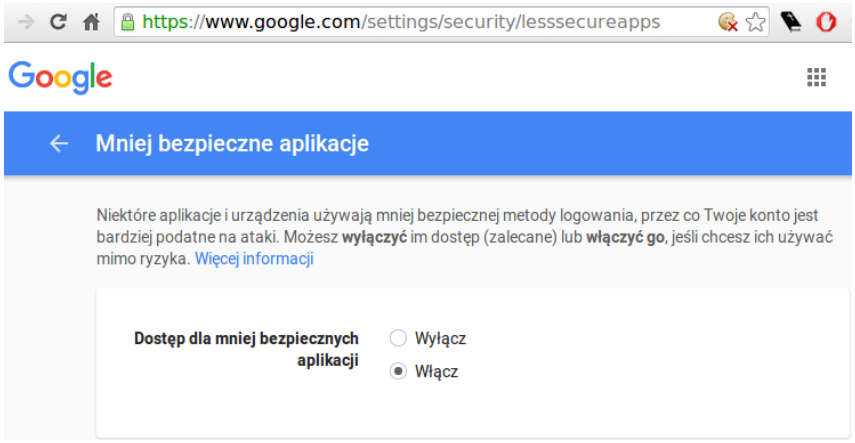
Używanie gmail'a

Gmail'a może działać od razu albo może wymagać dodatkowych ustawień:

- Gmail oczekuje prawdziwego użytkownika, nie robota; wykorzystuje różne heurystyki aby zapobiec logowaniu przez roboty,
- Gmail ma opcję `less secure`; https://support.google.com/accounts/answer/6010255?hl=pl, która pozwala na dostęp każdemu kto tylko zna hasło,
- może być potrzeba również wypełnienie kodu weryfikacyjnego (Captcha) zanim będziemy mogli wysyłać maile: https://accounts.google.com/b/0/displayunlockcaptcha
- Gmail zastępuje też zawsze nadawcę wiadomości danymi uwierzytelnionego użytkownika,
- w celu uniknięcia problemów z logowaniem należy użyć autoryzacji przez token (OAuth2) albo innego dostawcy.

Używanie gmail'a - wyłączenie `less secure`

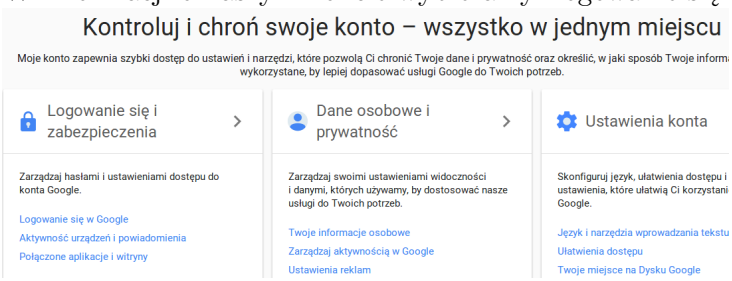
Opcję `less secure` można ustawić tutaj: https://www.google.com/settings/security/lesssecureapps



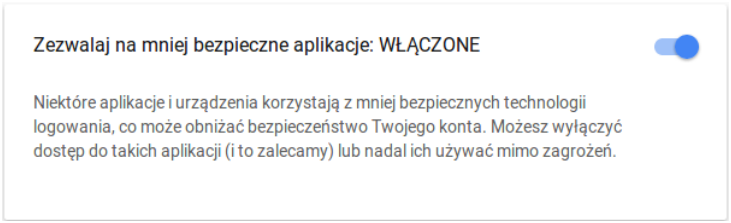
Używanie gmail'a - wyłączenie `less secure`

Opcja `less secure` w koncie google:

W informacji o naszym koncie wybieramy Logowanie się i zabezpieczenia:

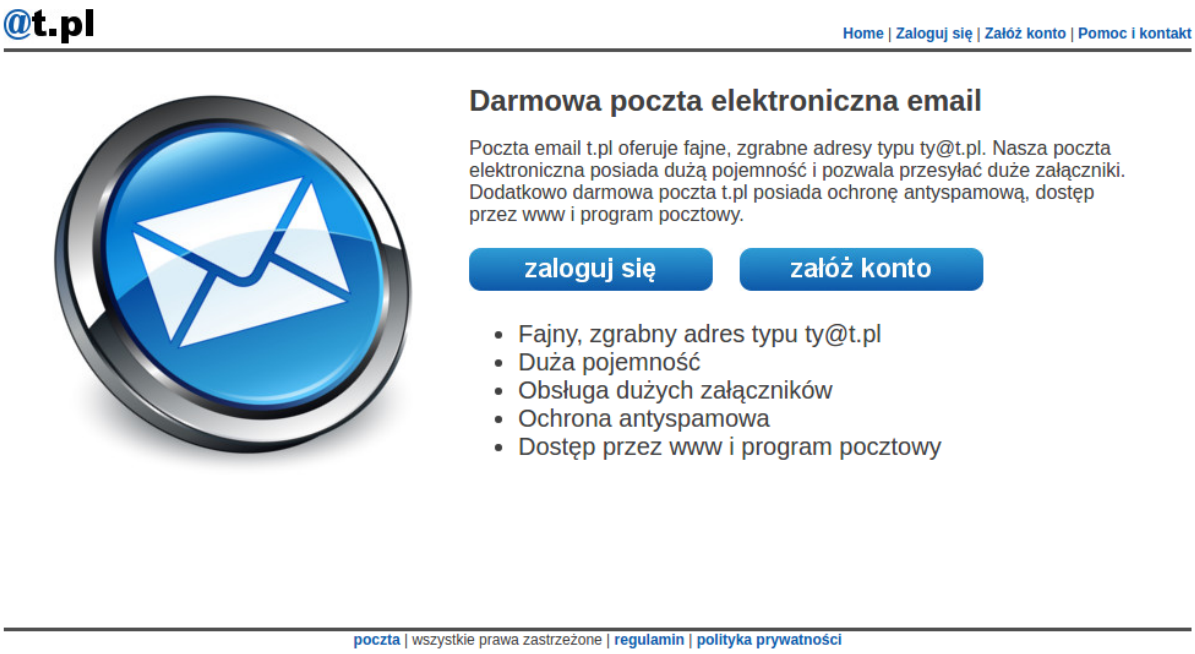


Opcja na samym dole:



1.3 t.pl

Inna poczta - t.pl



Darmowa poczta elektroniczna ze strony t.pl

Przydkład dla poczty t.pl - warstwa transportowa

Przygotowanie warstwy transportowej:

utils/mailerTransporter.js

```
...
// Serwer poczty przychodzącej: POP3, t.pl STARTTLS
// Serwer poczty wychodzącej: SMTP, t.pl, bez szyfrowania
let selfSignedConfig = {
  host: 't.pl',
  port: 465,
  secure: true, // używa TLS
  auth: {
    user: 'web20@t.pl', pass: '********'
  },
  tls: {
    // nie przerywa przy błędnym certyfikacie
    rejectUnauthorized: false
  }
};
let tTransporter = nodemailer.createTransport(selfSignedConfig);

module.exports = { gmailTransporter: gmailTransporter,
                   tTransporter: tTransporter };
```

Przydkład dla poczty t.pl - wysyłanie

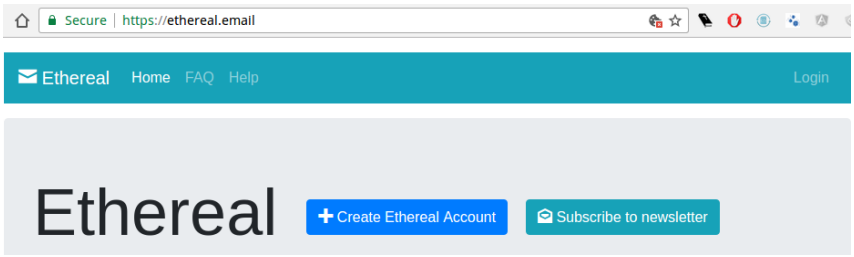
Wysyłanie wiadomości:

routes/index.js

```
router.get('/send-email3', function(req, res, next) {
  let mailOptions = {
      from: '"Student Web 2.0" <web20@t.pl>',     // adres nadawcy
      to: 'asdf@asdf.com, jkowalski@icis.pcz.pl', // lista odbiorców
      subject: 'Testowa wiadomość',               // temat wiadomości
      text: 'Treść wiadomości jako czysty tekst', // treść wiadomości tekstowej
                                                  // treść w HTML
      html: `<b>Treść wiadomości jak HTML</b>
      <p>Jakieś znaki w unicode, grecki alfabet: α, β, γ, ...</p>`
  };
  // wysyła maila dla ustawionej warstwy transportowej dla danych opcji
  tTransporter.sendMail(mailOptions, (error, info) => {
      if (error) {
          return console.log(error);
      }
      console.log('Wiadomość %s wysłana: %s', info.messageId, info.response);
      res.send('Wiadomość ' + info.messageId + ' wysłana: ' + info.response);
```

```
  });
});
```

Usługa Ethernal.email



Ethernal.email:

- sztuczna usługa SMTP, zwykle używana przez użytkowników Nodemailera
- pozwala za darmo "wysłać" fikcyjne maile i zobaczyć jak się prezentują
- pozwala to na zweryfikowanie treści i układu maila czy całej ich grupy
- wysłane/odebrane maile można przejrzeć dowolnym klientem email, również za pomocą przeglądarki

Usługa Ethernal.email I

Ethernal.email, konfiguracja warstwy transportowej:

utils/mailerTransporter.js

```
...

let eTransporter = nodemailer.createTransport({
    host: 'smtp.ethereal.email',
    port: 587,
    auth: {
        user: 'kpa6xtekmqbbohkw@ethereal.email',
        pass: 'FRuUkYStPeTgme6Xt9'
    }
});

module.exports = { gmailTransporter: gmailTransporter,
                   tTransporter: tTransporter,
                   eTransporter: eTransporter,
                   getTestMessageUrl: nodemailer.getTestMessageUrl };
```

Usługa Ethernal.email II

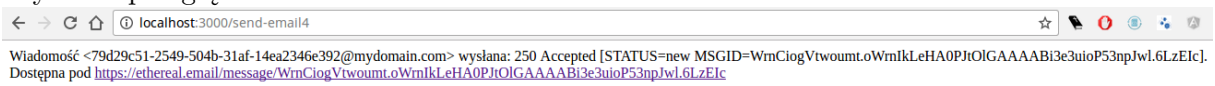
Ethernal.email, wysyłanie maila:

routes/index.js

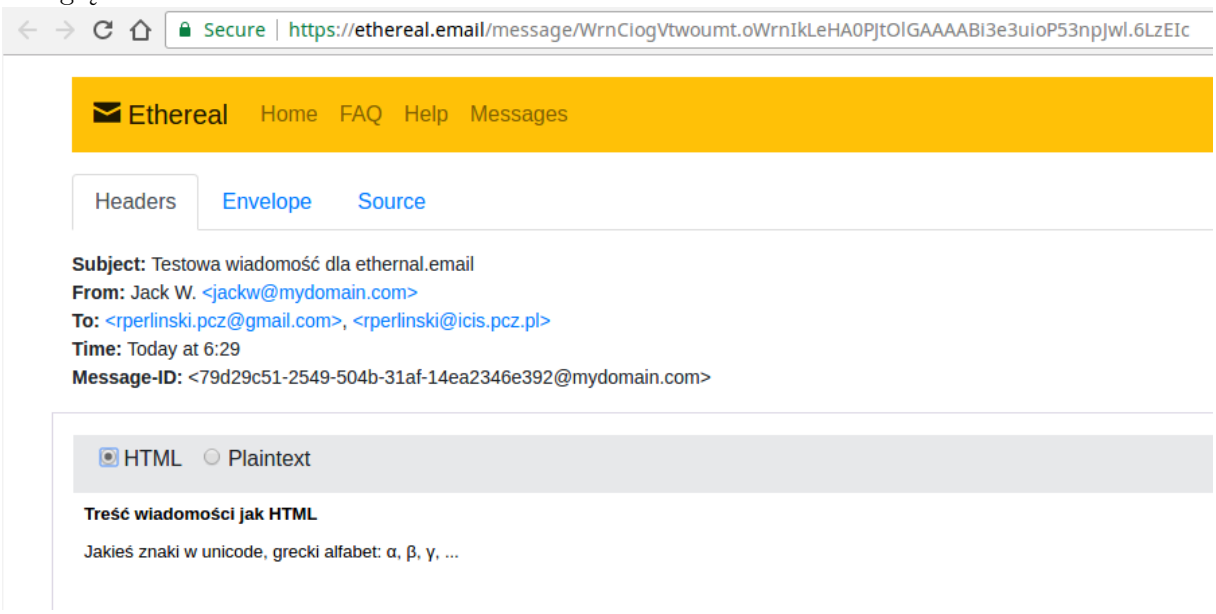
```
var eTransporter = require('../utils/mailerTransporter').eTransporter;
var getTestMessageUrl = require('../utils/mailerTransporter').getTestMessageUrl;
...

router.get('/send-email4', function(req, res, next) {
  let mailOptions = {
      from: '"Jack W." <jackw@mydomain.com>',        // adres nadawcy
      to: 'asdf@asdf.com, jkowalski@icis.pcz.pl',    // lista odbiorców
      subject: 'Testowa wiadomość - ethernet.email', // temat wiadomości
      text: 'Treść wiadomości jako czysty tekst',    // wiadomość tekstowa
      html: `<b>Treść wiadomości jak HTML</b>        // wiadomość HTML
      <p>Jakieś znaki w unicode, grecki alfabet: α, β, γ, ...</p>`
  };
  eTransporter.sendMail(mailOptions, (error, info) => {
    if (error) { return console.log(error); }
    var wiad = `Wiadomość $info.messageId wysłana: $info.response.<br> Dostępna
        pod <a href="$getTestMessageUrl(info)">$getTestMessageUrl(info)</a>`;
    res.send(wiad);
  });
});
```

Usługa Ethernal.email III

Wynik w przeglądarce

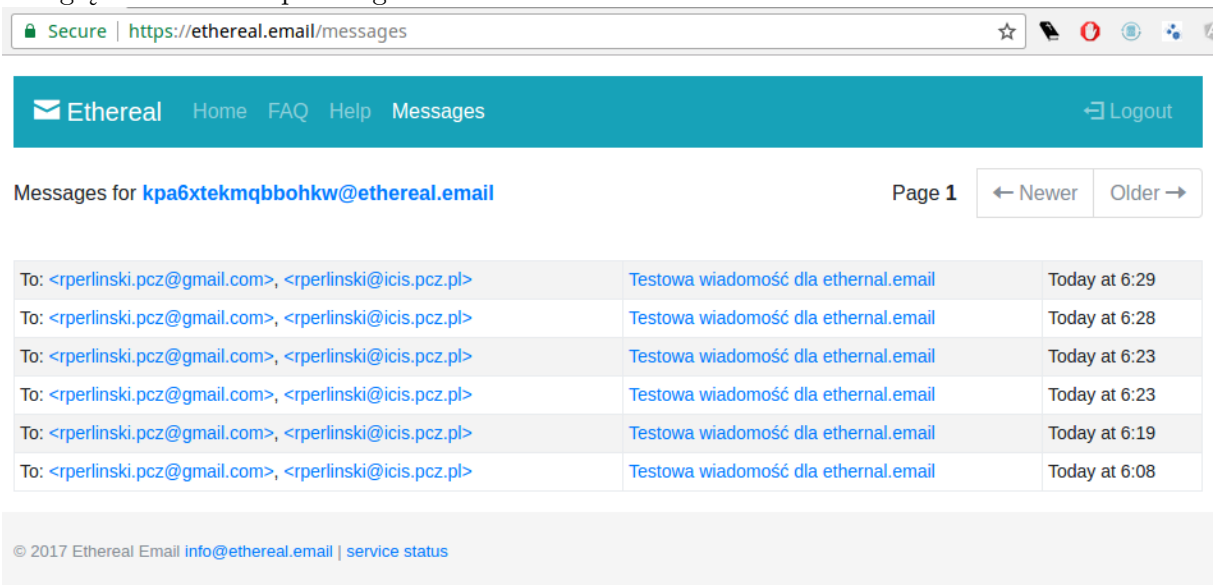


Przegląd maili online:



Usługa Ethernal.email IV

Przegląd maili online po zalogowaniu:



1.4 szablon, rejestracja i aktywacja konta

Tworzenie konta i jego aktywacja

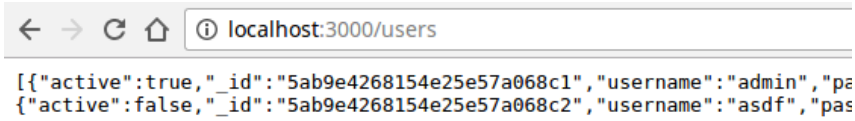
Wymagania w stosunku do aplikacji:

- formularz rejestracji zawierający przynajmniej **adres email** i **hasło**
- wysyłanie maila z linkiem aktywacyjnym do użytkownika po jego rejestracji
- link aktywacyjny aktywuje konto użytkownika
- użytkownik może się logować, korzystać z dodatkowych uprawnień

utils/db.pug

```
...
// schemat dokumentu opisującego użytkowników w kolekcji users
var userSchema = new mongoose.Schema({
  username: { type: String, required: true, unique: true},
  password: { type: String, required: true},
  email: { type: String, required: true, unique: true},
  active: {type: Boolean, default: false}
});
```

Dwóch domyślnie utworzonych użytkowników:



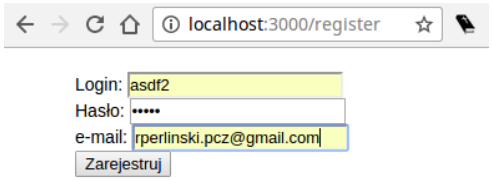
Formularz rejestracji

views/register.pug

```
block content
  form(action="/rejestracja", method="post")
    div
      label Login:
      input(type="text" name="username")
    div
      label Hasło:
      input(type="password" name="password")
    div
      label e-mail:
      input(type="email" name="email")
    div
      input(type="submit" value="Zarejestruj")
```

views/register.pug

```
router.get('/register', function(req, res) {
  res.render('register');
});
```



Rejestracja użytkownika - obsługa formularza

routes/index.js

```
router.post('/rejestracja', function(req, res) {
  let mailOptions = {
    from: '"MyApp Team" <web20@t.pl>', // adres nadawcy
    to: req.body.email,                // lista odbiorców
  };

  // tworzenie użytkownika i wysyłanie maila...
  var newUser = new User({"username":req.body.username, "password":sha1(req.body.password),
                          "email": req.body.email, "active": false});
  newUser.save(function(err,data) {
    if (err) return console.error(err);
    var host = "http://localhost:3000";
    // var host2 = "http://myapp.com";
    var aL = host + '/activate/' + data._id;

    res.render('mailBody', {activationLink: aL}, function(err, body){
      mailOptions.html = body;
      mailOptions.subject = 'Aktywacja konta';
      console.log(body);
      tTransporter.sendMail(mailOptions, (error, info) => {
        if (error) { return console.log(error); }
        res.send(‘
          <h3>Utworzony użytkownik:</h3> ${data}<br>
          Wiadomość o ID ${info.messageId} wysłana: ${info.response}.
        ‘);
      });
    });
  });
});
```

Treść maila generowana z szablonu

- Tworzymy szablon zawierający oczekiwaną treść maila, np. `mailBody.pug`.
- Renderujemy szablon i na jego zawartości wywołujemy funkcję, w której wyślemy maila.
- Wysyłamy maila zastępując treść wiadomości danymi zrenderowanymi z szablonu.

views/mailBody.pug

```
h3 Informacje
p
  | Dziękujemy za utworzenie konta na naszej stronie.
  | Poniżej znajdziesz link aktywacyjny do twojego konta.
h4 Aktywacja konta
p Link aktywacyjny:
  a(href=activationLink)= activationLink
p Your App Team &copy; 2018
```

Mail aktywacyjny - wysyłanie maila z szablonu

routes/index.js

```
router.get('/send-email2', function(req, res, next) {
  var aL = 'http://myapp.com/activate/36899d041ce8968435625b52b0ea827b5f08d645';

  res.render('mailBody', {activationLink: aL}, function(err, body){
    mailOptions.html = body;
    mailOptions.subject = 'Account activation';
    console.log(body);

    transporter.sendMail(mailOptions, (error, info) => {
        if (error) {
            return console.log(error);
        }
        console.log('Wiadomość %s wysłana: %s', info.messageId, info.response);
        res.send('Wiadomość ' + info.messageId + ' wysłana: ' + info.response);
    });
  });
});
```

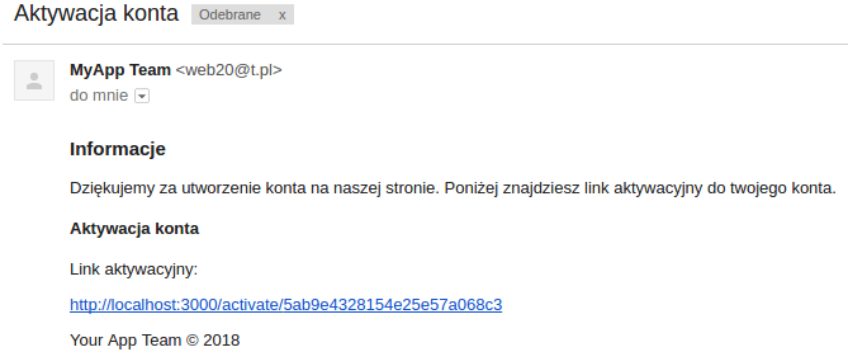
```
Wiadomość <126d04e2-6356-c356-71e2-19c0c7607983@gmail.com>
wysłana: 250 2.0.0 OK 1490148792 193sm3899286ljj.4 - gsmtp
```

Mail aktywacyjny - wynik

Treść maila:

```
<h3>Informacje</h3>
<p>Dziękujemy za utworzenie konta na naszej stronie.
   Poniżej znajdziesz link aktywacyjny do twojego konta.
</p>
<h4>Aktywacja konta</h4>
<p>Link aktywacyjny:
  <a href="http://localhost:3000/activate/5ab9e4328154e25e57a068c3">
          http://localhost:3000/activate/5ab9e4328154e25e57a068c3</a>
</p>
<p>Your App Team &copy; 2018</p>
```

Mail:



Rejestracja użytkownika - informacja w przeglądarce

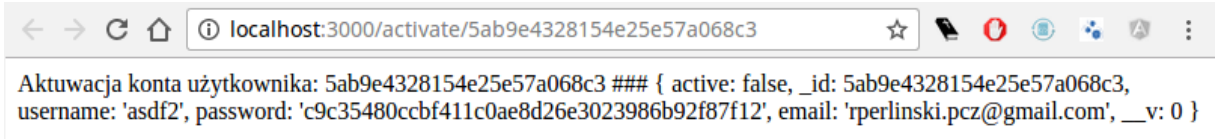
Wynik rejestracji używkownika w systemie:

Aktywacja konta użytkownika

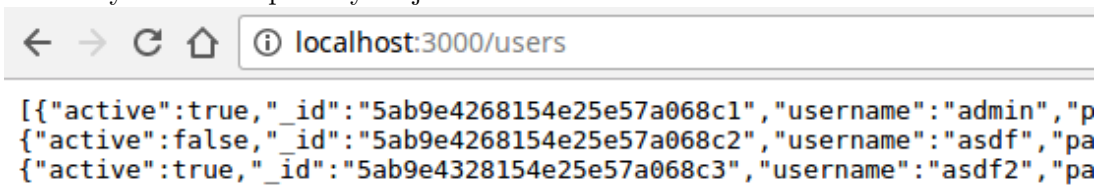
Kod aktywujący konto:

routes/index.js

```
// aktywowanie użytkownika o podanym w parametrze ID
router.get('/activate/:id', function(req, res, next) {
  User.findByIdAndUpdate( req.params.id, { $set:{active:true} },
                          {new: false}, function(err, data) {
    res.send("Aktywacja konta użytkownika: " + req.params.id + " ### " + data);
  });
});
```



Dane użytkowników po aktywacji konta:



Wymagane zmiany w uwierzytelnieniu

Dodatkowa metoda sprawdzająca uaktywnienie konta:

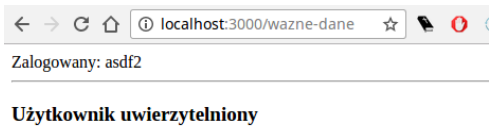
utils/db.js

```
...
userSchema.methods.accountActivate = function() {
  return this.active;
};
```

Mała zmiana w lokalnej stretegii uwierzytelnienia:

app.js

```
passport.use(new LocalStrategy(
  function(username, password, done) {
    User.findOne({ username: username }, function (err, user) {
      if (err) { return done(err); }
      if (!user) {
        return done(null, false, { message: 'Incorrect username.' });
      }
      if (!user.validPassword(password)) {
        return done(null, false, { message: 'Incorrect password.' });
      }
      if (!user.accountActivate()) {
        return done(null, false, { message: 'Account not activated.' });
      }
      return done(null, user);
    });
  }
));
```



2 Źródła

Źródła

- http://mongoosejs.com/
- https://nodemailer.com
- https://www.google.com/gmail/
- http://t.pl/
- https://blog.ragingflame.co.za/2012/6/28/simple-form-handling-with-express-and-nodemailer
- http://www.ryanray.me/sending-emails-with-jade-node-js-and-nodemailer