

IDENTYFIKACJA, UWIERZYTELNIANIE,
AUTORYZACJA, PRZYKŁAD LOGOWANIE DO
KONTA GOOGLE

Franciszek Sułowski

UŻYTE TECHNOLOGIE

- Node.js
- Express.js
- MongoDB (Atlas)
- Passport.js

KONFIGURACJA GOOGLE CLOUD CONSOLE

Rozpocznij bezpłatny okres próbny ze środkami w wysokości 300 USD. Bez obaw – gdy środki się skończą, nie pobierzemy żadnych opłat. [Więcej informacji](#)

ODRZUĆ

ZACZNIJ

Google Cloud

inzynieria internetu strona ▾

Wyszukaj (/) zasoby, dokumenty, usługi i inne treści

Szukaj

2

?

Interfejsy API i usługi



Iden...

Włączone interfejsy API i usł...

Biblioteka

Dane logowania

OAuth consent screen

Zgody dotyczące użytkowan...

Wybierz projekt



NOWY PROJEKT

Wyszukaj projekty i foldery



OSTATNIE

OZNACZONE GWIAZDKĄ

WSZYSTKIE

Nazwa

Identyfikator



Auto...

Do stos...

Ident...

http://...

+ D...

Auto...

Do stos...

Ident...

http://...

ANULUJ

+ DODAJ URI

DODAWANIE API DO PROJEKTU

The screenshot shows the Google Cloud Platform dashboard for a project named "My Project 3483". The left sidebar includes links for "Usługi Cloud", "Usługi i rozwiązania", "RZYPKIĘTE", "Interfejsy API i usługi" (which is highlighted with a red box), "Płatności", "Administracja", and "Marketplace". The main content area is titled "Interfejsy API i usługi" and contains sections for "Ruch", "Włączone interfejsy API i usługi" (with a red box around it), "Biblioteka" (which is circled in red), "Dane logowania", "OAuth consent screen", and "Zgody dotyczące użytkowania str".

This screenshot shows the "Google+ API" page from the Google Cloud API library. It features a summary card with the API name, developer (Google), and a brief description: "The Google+ API enables developers to build on top of the Google+ platform." To the right, there's a "Sieci społecznościowe" section with a "Blogger API" card. At the bottom, there are "Reklama" and "Gmail" cards.

API	Developer	Description
Google+ API	Google	The Google+ API enables developers to build on top of the Google+ platform.
Blogger API	Google	The Blogger API provides access to posts, comments and the Blogger blog.

TWORZENIE DANYCH LOGOWANIA

 Rozpocznij bezpłatny okres próbny ze środkami w wysokości 300 USD. Bez obaw – gdy środki się skończą, nie pobierzemy żadnych opłat. [Więcej](#)

 My Project 3483 ▾ Wyszukaj (/) zasoby, dokumenty, usługi i inne treści

API Interfejsy API i usługi

- ❖ Włączone interfejsy API i usługi
- ☰ Biblioteka
- 钥 Dane logowania**
- ☰ OAuth consent screen
- ⚙️ Zgody dotyczące użytkowan...

Dane logowania

+ UTWÓRZ DANE LOGOWANIA  

Klucz interfejsu API
Identyfikuje projekt za pomocą prostego klucza interfejsu API i dostęp

Identyfikator klienta OAuth
Pyta użytkownika o zgodę na dostęp aplikacji do jego danych

Konto usługi
Umożliwia uwierzytelnianie od serwera do serwera na poziom użyciu kont robota

Pomóż mi wybrać
Zadaje kilka pytań, które pozwalają zdecydować o typie użytkownika logowania

Nazwa
Brak kluczy API do wyświetlenia

Typ aplikacji *
Aplikacja internetowa

Nazwa *
Klient internetowy 1

Nazwa klienta OAuth 2.0. Ta nazwa służy tylko do identyfikacji klienta w konsoli i nie będzie widoczna dla użytkowników.

ⓘ Domeny identyfikatorów URI dodane poniżej zostaną automatycznie dodane do [ekranu zgody OAuth](#) jako [domeny autoryzowane](#).

Autoryzowane źródła JavaScriptu ?

Do stosowania z żądaniami z przeglądarki

Identyfikatory URI 1 *
http://localhost:3000

+ DODAJ URI

Autoryzowane identyfikatory URI przekierowania ?

Do stosowania z żądaniami z serwera WWW

Identyfikatory URI 1 *
http://localhost:3000/auth/google/callback

url naszej strony

url do przekierowania po udanym zalogowaniu

Klient OAuth został utworzony

Identyfikator klienta i klucz tajny zawsze są dostępne na stronie Dane logowania w sekcji Interfejsy API i usługi

 Dostęp OAuth jest ograniczony do [użytkowników testowych](#) wymienionych na [ekranie zgody OAuth](#).

Identyfikator klienta

425...
5dqc...@accounts.google.com
googleusercontent.com 

Tajny klucz klienta

G...X-
g_k...@accounts...Z9eloJ3_Vp 

Data utworzenia

22 listopada 2023 19:16:55 GMT+1

Stan

 Włączono

 [POBIERZ JSON](#)

Ekran zgody OAuth

Użytkownicy testowi

 [ADD USERS](#)

 [Filtruj](#) Wpisz nazwę lub wartość właściwości

Informacje o użytkownikach

francishekk@gmail.com 

BACKEND

Łączenie autoryzacji Google z naszą stroną

Instalacja niezbędnych pakietów do strony

```
bash
```

```
npm install express mongoose passport passport-google-oauth20
```

Łączenie z bazą danych Atlas

javascript

```
mongoose.connect('YOUR_MONGODB_ATLAS_CONNECTION_STRING', {  
  useNewUrlParser: true,  
  useUnifiedTopology: true,  
});  
  
const User = mongoose.model('User', {  
  googleId: String,  
  displayName: String,  
  // Other fields as needed  
});
```

Konfiguracja Passporta oraz Google Strategy i OAuth 2

javascript

```
const passport = require('passport');
const GoogleStrategy = require('passport-google-oauth20').Strategy;

passport.use(new GoogleStrategy({
    clientID: 'YOUR_GOOGLE_CLIENT_ID',
    clientSecret: 'YOUR_GOOGLE_CLIENT_SECRET',
    callbackURL: 'http://localhost:3000/auth/google/callback'
}, async (accessToken, refreshToken, profile, done) => {
    // User authentication logic here
}));
```

Konfiguracja ścieżek

```
// ściezka do strony domowej
app.get('/', (req, res) => {
  res.send('Welcome to the home page!');
});

// ściezka do podstrony logowania google
app.get('/auth/google', passport.authenticate('google', { scope: ['profile', 'email'] }));

// ściezka do przekierowania w razie nieudanego logowania
app.get('/auth/google/callback', passport.authenticate('google', { failureRedirect: '/' }), (req,
res) => {
  res.redirect('/profile');
});
```

```
// ściezka do profilu --> dostępna po logowaniu
app.get('/profile', (req, res) => {
  if (req.isAuthenticated()) {
    res.send(`Welcome, ${req.user.displayName}!`);
  } else {
    res.redirect('/');
  }
});

// ściezka wylogowania
app.get('/logout', (req, res) => {
  req.logout();
  res.redirect('/');
});
```

 Zaloguj się przez Google

Wybierz konto

by przejść do aplikacji [test1](#)



 Franek Sułowski
francishekk@gmail.com



Wylogowano



Wylogowano



Wylogowano



Wylogowano



Wylogowano

 Użyj innego konta

Aby można było przejść dalej, Google udostępnia aplikacji test1 Twoją nazwę użytkownika, adres e-mail, ustawienia języka i zdjęcie profilowe.

Atlas francishekk'... Access Manager Billing

Project 0 Data Services App Services Charts

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 317B TOTAL DOCUMENTS

Find Indexes Schema Anti-Patterns 0

mywebsite users

Filter Type a query: { field: 'value' }

email: "dfsfd fds@gmail.com"
password: "dsasadsada"
_id: 0

_id: ObjectId('653a98c5836cc0a6371bdd72')
username: "dasdsa"
email: "adssad@gmail.com"
password: "dsadasdas"
_id: 0


_id: ObjectId('655e3c18454ecbae9ce60aa0')
googleId: "103106020298241150022"
displayName: "Franek Sułowski"
_id: 0

System Status: All Good

KONIEC

Franciszek Sułowski

Menedżery haseł

Dawid Mikoś

Edukacja techniczno-informatyczna

Akademia Górnictwo-Hutnicza im. Stanisława Staszica w Krakowie
AGH University of Krakow

Menedżer haseł – co to?

Menedżer haseł to aplikacja, która generuje losowe i unikatowe hasła oraz przechowuje je w bezpieczny, szyfrowany sposób w pamięci komputera lub telefonu. Wykorzystuje do tego zaszyfrowaną bazę danych. Znając hasło główne użytkownik ma dostęp do całej bazy.

Menedżery haseł – jak działają?

Menedżer haseł może być oferowany w formie:

- **programu instalowanego na komputerze** (Windows, macOS, Linux)
- **aplikacji na telefon lub tablet** (iOS, Android)
- **usługi działającej 100% online** (przez stronę WWW)
- **wtyczki/dodatku do przeglądarki** (Chrome, Firefox, Brave, Opera, Edge, Safari itd.)

Podstawowe funkcje managera haseł

- Generowanie haseł
- Auto wpisywanie loginów i haseł do formularzy
- Przechowywanie haseł w bezpieczny sposób

Dodatkowe funkcje managera haseł

- Ochrona przed phishingiem
- Sprawdzanie wycieków haseł
- Kody jednorazowe 2FA

Przykłady

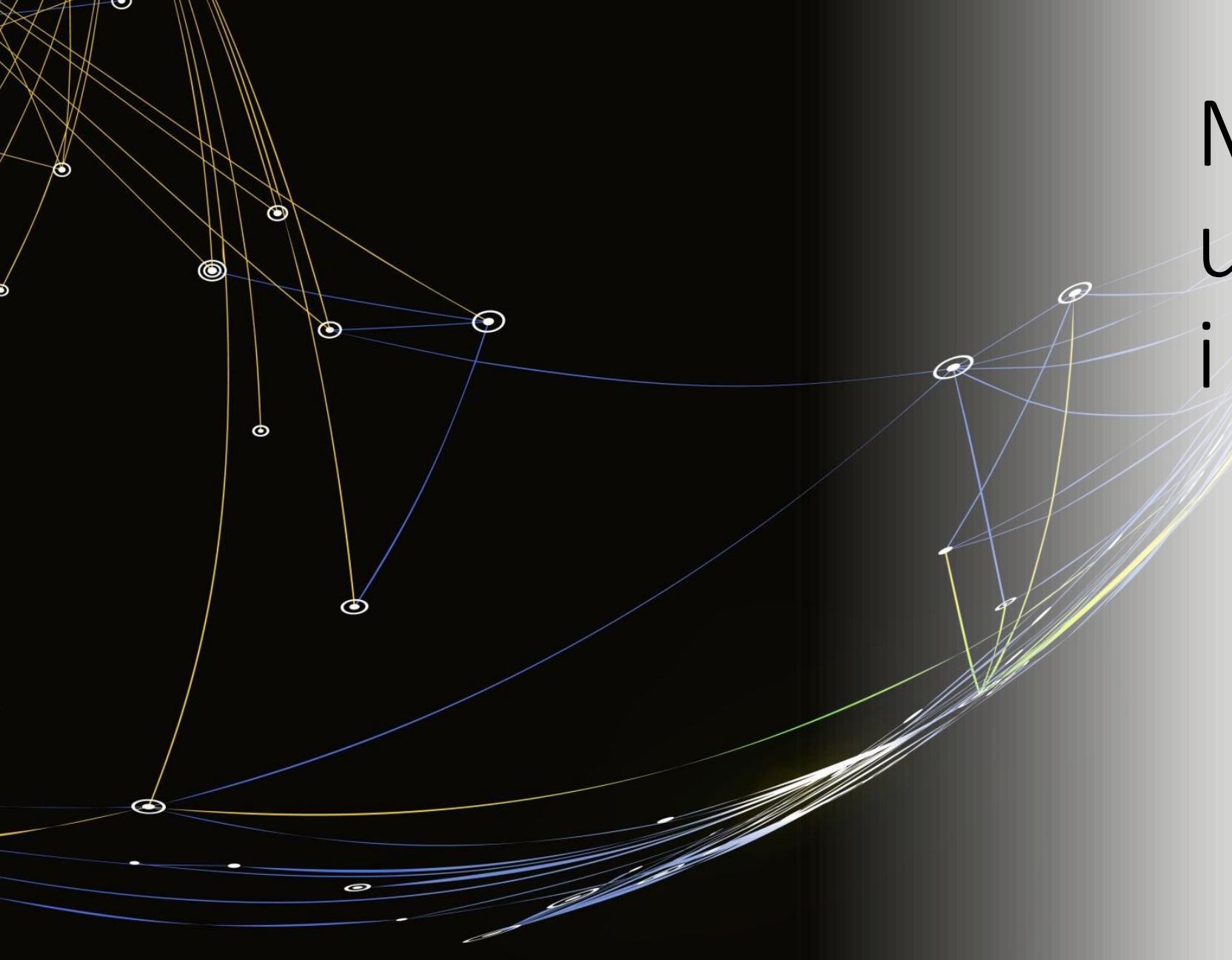
- 1Password
- LastPass (popularna usługa w chmurze)
- RoboForm (wygodny, multiplatformowy program)
- NordPass (narzędzie od producenta popularnego VPN-a)
- Keeper (nagradzany menedżer – od użycia personalnego do oferty dla korporacji)
- BitWarden (open source, w chmurze producenta lub własnej)
- Dashlane (kolejna popularna chmurowa usługa)
- KeePassXC (open source i działający lokalnie)

Dziękuję za uwagę

Najbezpieczniejsze jest hasło, którego nie znasz!

Bibliografia

- https://pl.wikipedia.org/wiki/Mened%BCer_hase%C5%82
- <https://kwestiabezpieczenia.pl/manedzer-hasel/>
- (dostęp 29.11.2023)



Metody uwierzytelniania i autoryzacji

Norbert Plesner ETI, Gr.2

Co to jest uwierzytelnianie?

Jest to proces polegający na potwierdzaniu zadeklarowanej tożsamości podmiotu biorącego udział w procesie komunikacji.



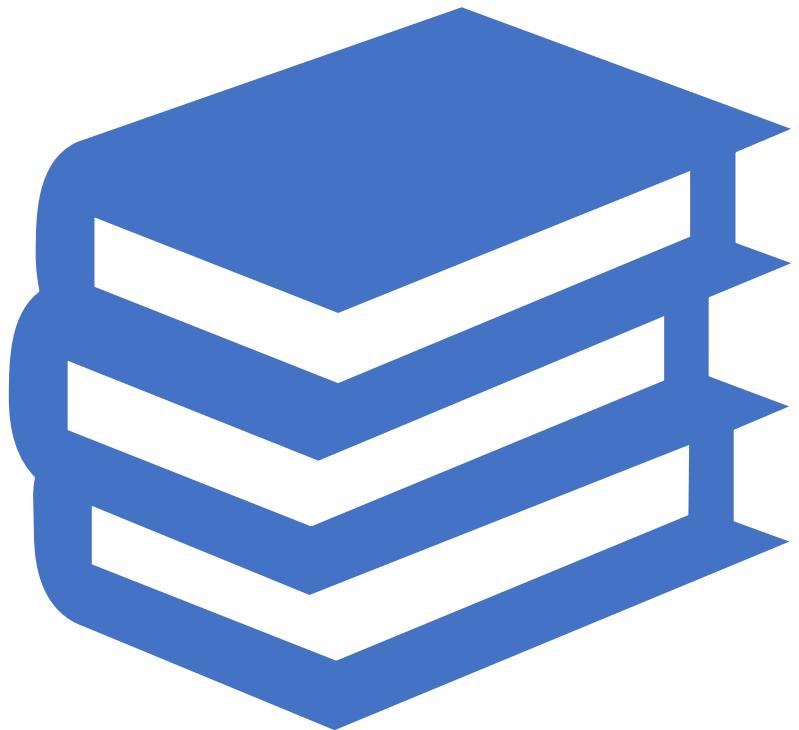


Historia
uwierzytelniania

Co to jest autoryzacja?

Jest to proces przyznania
zezwolenia na wykonywanie
określonych czynności.



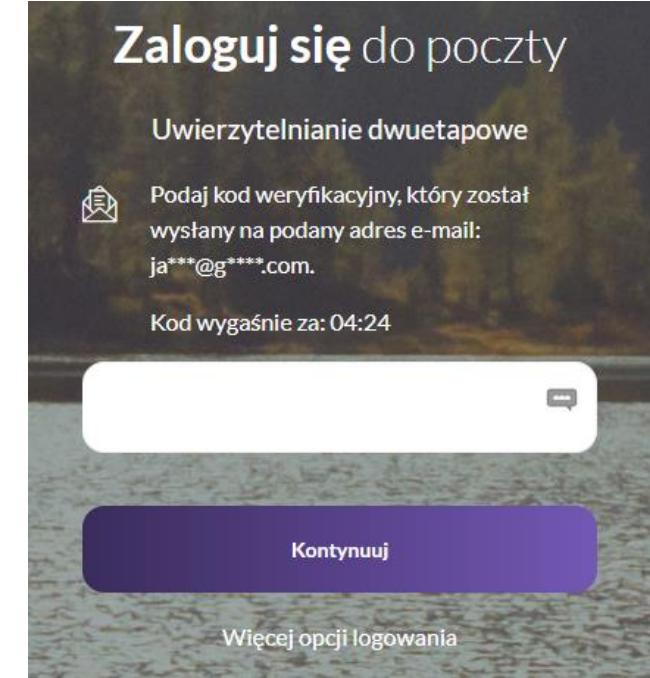


Historia autoryzacji

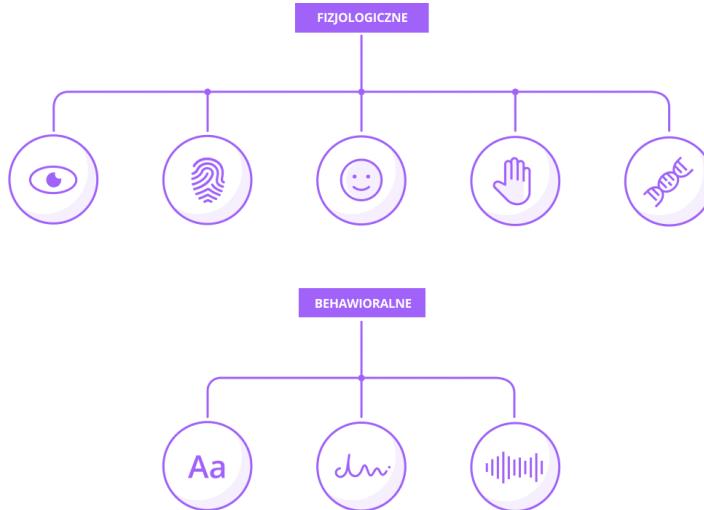
Metody uwierzytelniania

- hasła
- e-mail
- sms
- biometria
- bankowość elektroniczna
- wideo weryfikacja
- karty i tokeny





Zilustrowane przykłady



Zilustrowane przykłady cd.

Zaawansowane metody uwierzytelniania

- Uwierzytelnianie dwuskładnikowe 2FA
- Uwierzytelnianie wieloskładnikowe MFA



Metody autoryzacji



- uwierzytelnianie jako autoryzacja



- listy kontroli dostępu



- kontrola dostępu oparta na rolach



- kontrola dostępu oparta na atrybutach



Uwierzytelnianie,
a autoryzacja

uwierzytelnianie ≠ autoryzacja

Zagrożenia związane z działaniami cyberprzestępców

- phishing

- ataki ransomware

- ataki na systemy IoT (Internet of Things)

- ataki DDoS (Distributed Denial of Service)

- kradzież tożsamości

Dziękuję za uwagę

Bibliografia:

- <https://dopodpisu.pl/metody-uwierzytelniania-osoby-fizycznej-przewodnik-poroznych-technikach/>
- <https://instytutcyber.pl/encyklopedia/autoryzacja/>
- <https://pl.wikipedia.org/wiki/Uwierzytelnianie>
- https://sebitu.pl/na-czym-polega-uwierzytelnienie-wieloskladnikowe-mfa/#Czym_jest_uwierzytelnianie_wieloskladnikowe_MFA
- <https://www.gov.pl/web/baza-wiedzy/konfigurowanie-uwierzytelniania-dwuskladnikowego-2fa>
- <https://learn.microsoft.com/pl-pl/entra/identity-platform/authorization-basics>
- <https://www.netcomplex.pl/zdalny-dostep-do-sieci-autoryzacja>
- <https://www.kozminski.edu.pl/pl/review/najwazniejsze-wyzwania-zwiazane-z-cybersecurity>



Zalety i wady JWT (JSON Web Tokens)



Krzysztof Żółtek

Zalety

Tworzenie i weryfikacja tokenów "w locie"

Prosta implementacja i szybszy czas rozwoju

Zastosowanie w różnych usługach

Opiera się na standardzie JSON

Wady

Rozmiar Tokena

Brak automatycznego odświeżania

Bezpieczeństwo podpisów:

Trudność w odwołaniu tokenów:

Podatność na błędy implementacji:

Wysoki próg wejścia:

Brak możliwości wylogowania lub unieważnienia sesji:

Bibliografia

- <https://www.thegeekyway.com/pros-and-cons-of-jwt/>
- <https://www.bdabek.pl/tokeny-jwt-a-zarzadzanie-sesja-✓/>
- <https://highload.today/jwt-auth-json/>

Dziękuję za uwagę



{

Uwierzytelnianie i autoryzacja API z passport.js JSON Web Token

< Patryk Urbański >

...

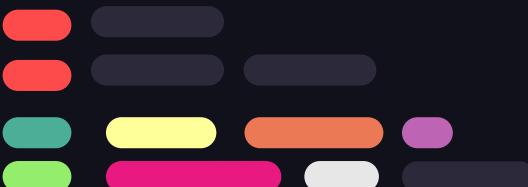
}

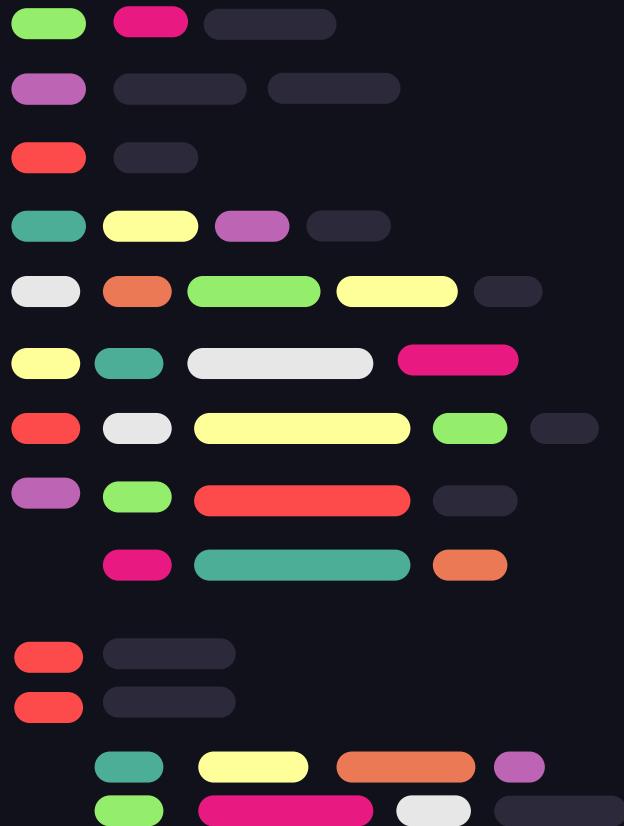
Passport.js

Biblioteka do uwierzytelniania dla aplikacji opartych na platformie Node.js. Jej głównym celem jest ułatwienie implementacji strategii uwierzytelniania.

Cechy:

- modularyzacja,
- łatwość użycia,
- Wsparcie dla różnych strategii,
- Middleware,
- rozszerzalność





{ . . .

server.js

} . . .

Import plików i bibliotek

```
//import plików i bibliotek
const express = require('express');
const bodyParser = require('body-parser');
const passport = require('passport');
const ExtractJwt = require('passport-jwt').ExtractJwt;
const LocalStrategy = require('passport-local').Strategy;
const JwtStrategy = require('passport-jwt').Strategy;
const jwt = require('jsonwebtoken');

const app = express();
const port = 3000;
```



- Express: framework do tworzenia aplikacji webowych w Node.js,
- Body-parser: middleware do obsługi danych przesyłanych w ciele żądania,
- Passport: Biblioteka do obsługi uwierzytelniania w aplikacjach Node.js,
- Passport-local: Strategia uwierzytelniania dla passport.js,
- Passport-jwt: strategia uwierzytelniania dla passport.js,
- jwt : biblioteka do generowania i weryfikowania tokenów JWT.

Przykładowa baza danych użytkowników

- Users: tablica reprezentująca przykładową bazę danych użytkowników.



```
//przykładowa baza danych użytkowników
const users = [
  { id: 1, username: 'testuser', password: 'testpassword' },
];
```

Konfiguracja passport.js z użyciem strategii lokalnej

Tworzenie strategii lokalnej passport.js, która sprawdza, czy podane dane użytkownika są zgodne z danymi w bazie.



```
//konfiguracja passport.js z użyciem strategii lokalnej
passport.use(new LocalStrategy(
  {usernameField: 'username', passwordField: 'password'},
  (username, password, done) => {
    const user = users.find(u => u.username === username && u.password === password);
    if (user) {
      return done(null, user);
    } else {
      return done(null, false, {message: "Invalid username or password"});
    }
  }
));
```

Konfiguracja passport.js z użyciem strategii JWT

```
//konfiguracja passport.js z użyciem strategii JWT
const jwtOptions = {
    jwtFromRequest: ExtractJwt.fromAuthHeaderAsBearerToken(),
    secretOrKey: "key",
};

passport.use(new JwtStrategy(jwtOptions, (payload, done) => {
    const user = users.find(u => u.id === payload.id);
    if (user) {
        return done(null, user);
    } else {
        return done(null, false);
    }
}));
```

Tworzenie strategii JWT dla passport.js, która weryfikuje token JWT.



Konfiguracja middleware

```
//konfiguracja middleware  
app.use(bodyParser.json());  
app.use(passport.initialize());  
app.use(express.static('public'));
```

- Ustawienie middleware `body-parser` , który umożliwia obsługę danych w formacie JSON,
- Ustawienie middleware `passport.initialize()`. Które inicjalizuje passport.js.



Trasa rejestracji

```
//trasa rejestracji
app.post('/register', (req, res) => {
  const { username, password } = req.body;
  const newUser = { id: users.length + 1, username, password };
  users.push(newUser);
  res.json({message: "registration successful", user: newUser});
})
```



- Wyciągnięcie wartości `username` i `password` z obiektu `req.body` który zawiera dane z formularza rejestracyjnego,
- Stworzenie nowego obiektu `newUser` oraz przypisuje mu unikalne ID oraz pozyskane wcześniej z `req.body` nazwę i hasło,
- Wyświetlenie informacji że rejestracja udała się poprawnie,

Trasa logowania

```
//trasa logowania
app.post('/login', passport.authenticate('local', {session: false }), (req, res) => {
  const token = jwt.sign({ id: req.user.id }, jwtOptions.secretOrKey);
  res.json({ message: 'login successful', token });
});
```

- Po pomyślnym uwierzytelnieniu generowany jest token JWT przy użyciu funkcji `jwt.sign()`, pobierany jest on z obiektu `req.body` ,
- `jwtOptions.secretOrKey` to sekret używany do przypisywania tokenu,
- Odpowiedz na żądanie klienta, wysyłając odpowiedź w formacie JSON. Komunikat informuje klienta o pomyślnym zalogowaniu a także zawiera wygenerowany token JWT.



Zabezpieczona trasa



Funkcja odpowiada na żądanie klienta, wysyłając odpowiedź w formacie JSON. Komunikat informuje klienta o uzyskanym dostępie do bezpiecznej trasy, a także zawiera dane użytkownika, które zostały dostarczone przez passport.js po pomyślnym uwierzytelnieniu.

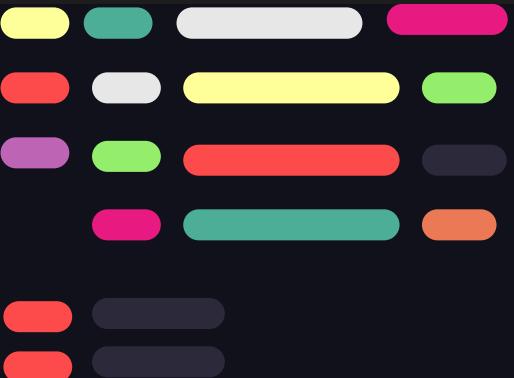
```
//zabezpieczona trasa
app.get('/secure-route', passport.authenticate('jwt', { session: false }), (req, res) => {
  res.json({ message: "access granted to secure route", user: req.user });
});
```

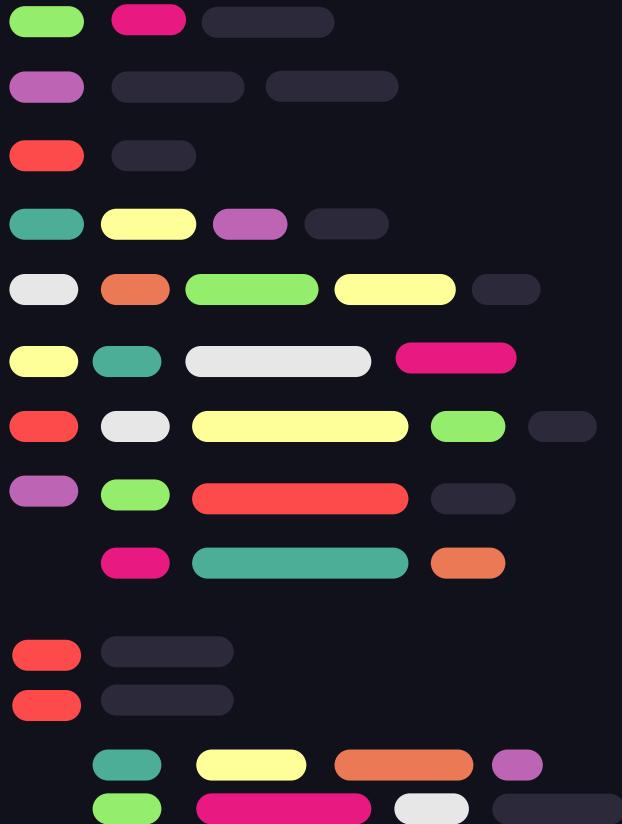
Uruchomienie serwera

Ustawienie serwera na słuchanie oraz aktywacja serwera na określonym wcześniej porcie.



```
//uruchomienie serwera
app.listen(port, () => {
  console.log(`server is running on http://localhost:\${port}`);
});
```





{ . .



index.html

} . .

Html head

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Passport.js JWT Example</title>

  <!--style strony-->
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 40px;
    }

    label, input, button {
      margin-bottom: 10px;
      display: block;
    }

    button {
      padding: 8px;
      cursor: pointer;
    }

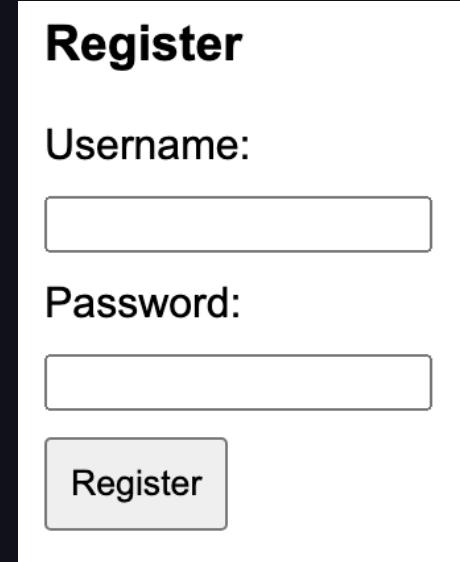
    #secureContent {
      margin-top: 20px;
    }
  </style>
</head>
```

-standardowy nagłówek strony html,
-style css tworzące wygląd strony.



Formularz rejestracji

```
<!--formularz rejestracji-->
<div id="registerForm">
  <h3>Register</h3>
  <label for="regUsername">Username:</label>
  <input type="text" id="regUsername" required>
  <label for="regPassword">Password:</label>
  <input type="password" id="regPassword" required>
  <button onclick="register()">Register</button>
</div>
```



The image shows a registration form interface. It features a title 'Register' at the top, followed by two input fields labeled 'Username:' and 'Password:', each with a corresponding text input box. Below these fields is a large 'Register' button.

Register

Username:

Password:

Register



Formularz logowania

```
<!--formularz logowania-->


<h3>Login</h3>
  <label for="loginUsername">Username:</label>
  <input type="text" id="loginUsername" required>
  <label for="loginPassword">Password:</label>
  <input type="password" id="loginPassword" required>
  <button onclick="login()">Login</button>


```

Login

Username:

Password:

Login



Trasa zabezpieczona



Secure Content

Access Secure Route

```
<!--trasa zabezpieczona-->
<div id="secureContent" style="display: none;">
  <h3>Secure Content</h3>
  <p id="secureMessage"></p>
  <button onclick="getSecureRoute()">Access Secure Route</button>
</div>
```

Funkcja register

```
//funkcja register
async function register() {
    const username = document.getElementById('regUsername').value;
    const password = document.getElementById('regPassword').value;

    const response = await fetch('http://localhost:3000/register', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
        },
        body: JSON.stringify({ username, password }),
    });

    const data = await response.json();
    alert(data.message);
}
```

- Pobranie wartości `username` i `password` z pól tekstowych i przypisuje je do zmiennych `username` i `password`,
- Wykorzystanie funkcji `fetch` do wysłania żądania na ścieżkę `/register`, następnie przesyła dane w formie obiektu JSON w ciele żądania,
- Wyświetlenie komunikatu z obiektu `data` w oknie alertu.



Funkcja login

```
//funkcja login
async function login() {
    const username = document.getElementById('loginUsername').value;
    const password = document.getElementById('loginPassword').value;

    const response = await fetch('http://localhost:3000/login', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
        },
        body: JSON.stringify({ username, password }),
    });

    const data = await response.json();
    alert(data.message);

    if (response.status === 200) {
        document.getElementById('secureContent').style.display = 'block';
    }
}
```

- Pobranie danych z okien tekstowych i umieszczenie ich w obiektach `username` i `password`
- Wykorzystanie funkcji `fetch` do wysłania żądania na ścieżkę `/login` i przesyła dane użytkownika w formie obiektu JSON,
- Wyświetlenie komunikatu zwrotnego z obiektu `data` ,
- Sprawdzenie czy uwierzytelnienie jest udane, jeśli tak wyświetla element `secureContent`



Funkcja bezpiecznej trasy

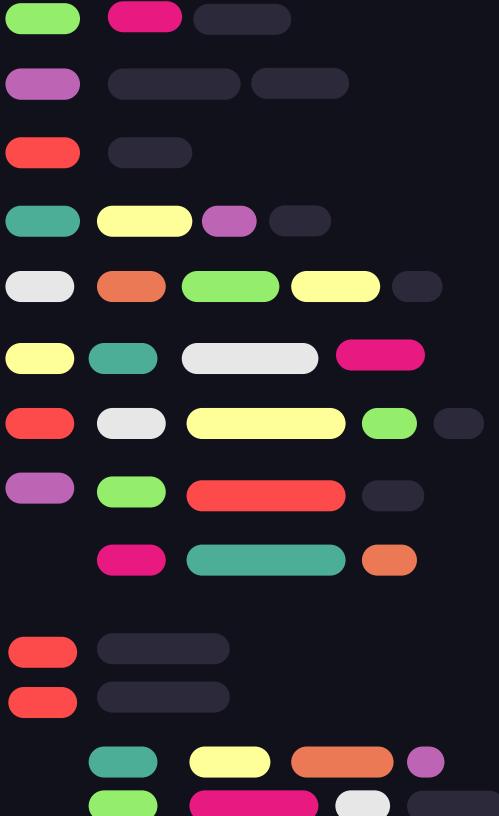
```
//funkcja secure route
async function getSecureRoute() {
    const token = prompt('Enter JWT Token:');
    if (!token) return;

    const response = await fetch('http://localhost:3000/secure-route', {
        method: 'GET',
        headers: {
            'Authorization': `Bearer ${token}`,
        },
    });

    const data = await response.json();
    document.getElementById('secureMessage').textContent = data.message;
}
```

- Wyświetlenie okna dialogowego do wpisania tokenu JWT i jeśli użytkownik anuluje okno żądanie nie jest wysyłane na serwer,
- Wykorzystanie funkcji fetch do wysłania żądania na trasę `/secure-route` i ustawia nagłówek żądania na nagłówek autoryzacyjny,
- Aktualizowanie zawartości o identyfikatorze `secureMessage`, umieszczając w nim komunikat zwrotny z obiektu `data`





Thanks !

CREDITS: This presentation template was created by **Slidesgo**, and includes icons by **Flaticon**, and infographics & images by **Freepik**

Please keep this slide for attribution