

# PassportJs

Von Florian Wiese

# Overview

- Für Node entwickelt
- Für Authentifikationsprozesse gemacht
- Jede weitere Funktionalität wird der Applikation überlassen
- Traditionell loggt man sich mit Username und Passwort ein
- Zu Zeiten der sozialen Medien kann man aber auch Facebook und Twitter zur Anmeldung nutzen

# Authenti- fizierung

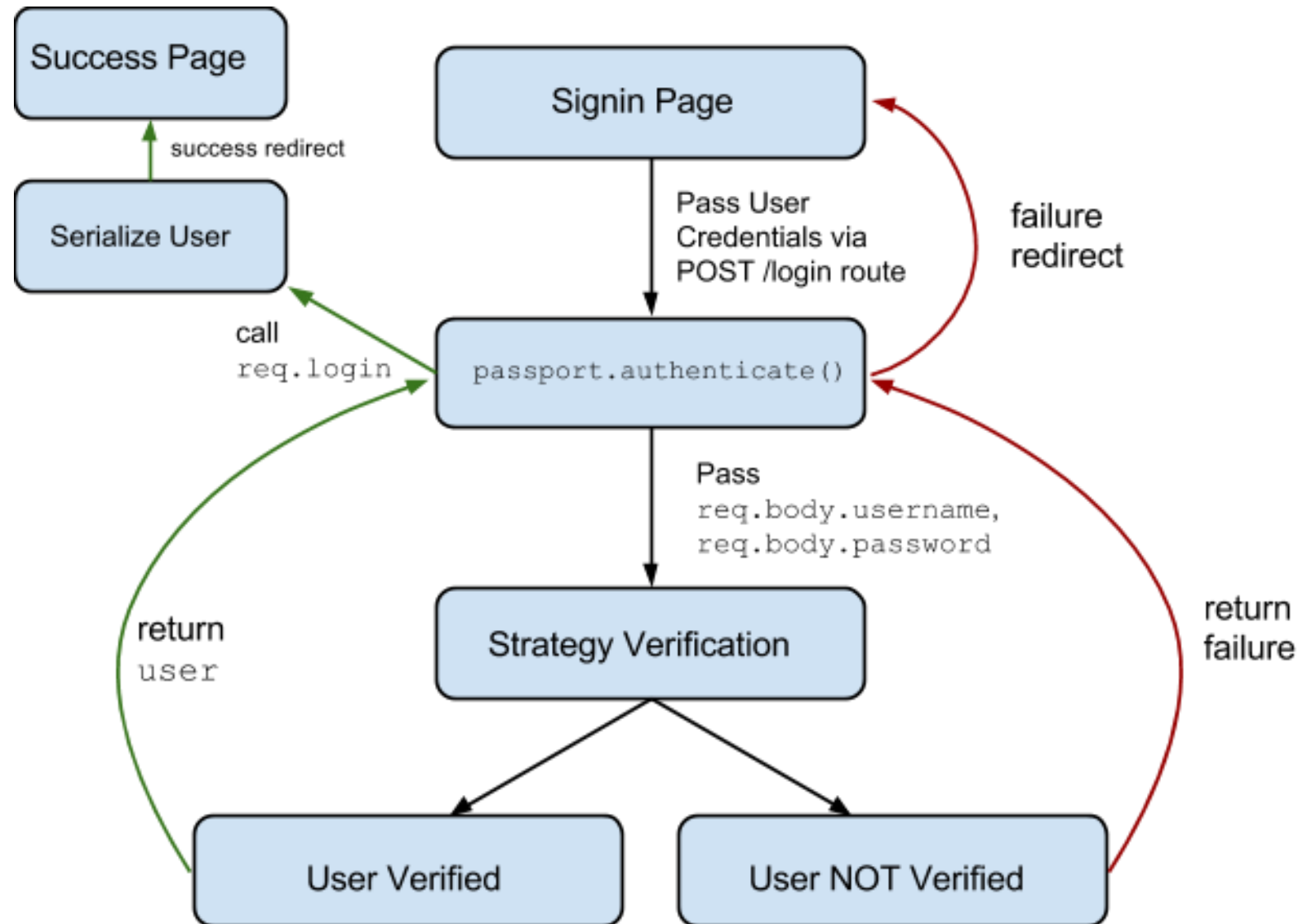
- Hierfür muss man nur die Funktion `passport.authenticate()`; aufrufen

```
app.post('/login',  
  passport.authenticate('local'),  
  function(req, res) {  
    // If this function gets called, authentication was successful.  
    // `req.user` contains the authenticated user.  
    res.redirect('/users/' + req.user.username);  
  });
```

# Authenti- fizierung

- Wenn die Authentifizierung jetzt fehl schlägt wird ein 401 Unauthorized Status gesendet und der Route\_Handler(URL Regler) leitet nicht weiter
- Falls doch wird man weitergeleitet und der User wird gesetzt
-

# Authenti- fizierung



# Strategien

- Für das einloggen kann man bei PassportJS verschiedene Strategien benutzen
- Diese muss man implementieren
- Dafür kann man Username und Passwort benutzen oder die oAuth oder OpenID

# Flash Nachrichten

- Bei diesen Weiterleitungen können sogenannte Flash Nachrichten aufploppen
- Diese zeigen Informationen für den User an
- Wie zum Beispiel: Invalid Username or password bei Fehlschlagen und zum Beispiel Willkommen für einen erfolgreichen Login

# Flash Nachrichten

The screenshot shows a web browser window with the address bar displaying 'localhost:3000/register'. The page title is 'Express-Stormpath (sample)'. The main heading is 'Flask-Express' with 'Home' and 'Login' links. The section is titled 'Create a New Account'. A note states: 'Registering for this site will create a new user account for you via Stormpath, then log you in automatically.' A blue box contains a note: 'NOTE: password must be between 8 and 100 characters, and must lowercase letters, uppercase letters, and numbers.' The registration form has two input fields: 'Email' and 'Password', each with a label to its left. Below the fields is an orange 'Register' button. At the bottom, a section titled 'A Note About Stormpath Security' explains that the application should be deployed behind SSL/TLS and that Stormpath user creation is secure.

Express-Stormpath (sample) X

localhost:3000/register

Flask-Express

Home Login

### Create a New Account

Registering for this site will create a new user account for you via Stormpath, then log you in automatically.

**NOTE:** password must be between 8 and 100 characters, and must lowercase letters, uppercase letters, and numbers.

Email

Password

Register

### A Note About Stormpath Security

In a real environment, you should **only deploy your application** behind SSL / TLS to avoid having user credentials sent in plain text over the network.

Stormpath user creation is incredibly secure, is not susceptible to any known vulnerabilities (MITM, hashing issues, replay attacks, etc.), and provides you with a number of options to improve security (including built-in email verification, among other things). We will not verify your account by email now, but this can be easily enabled.



# Strategien

- Die Kategorien:
- 1. Username und Passwort
- 2. OpenID
- 3. OAuth

# Username & Passwort

- Meist verbreitete Authentifizierungsstrategie
- Unterstützt durch ein Modul namens passport-local

# OpenID

- Mit OpenID kann man sich auf jeden unterstützten Website anmelden
- Dafür muss man sich einmal bei einem Open ID Provider anmelden
- Diese Daten kann man dann bei den entsprechenden Webseiten anwenden
- Für Phishing Anfälle anfällig, da man leicht eine Weiterleitung erstellen zu einer Website erstellen kann, die wie eine Provider Website aussieht
- Ein OpenID Provider ist zum Beispiel Google

# OAuth

- Der Nutzer kann dadurch Dritten erlauben auf seine Daten zugreifen
- Standardisierte, sichere API-Autorisierung
- Dabei gibt es dann vier Rollen:
- Ressource Owner : Er ist derjenige der Dritten Zugang zu seinen Daten gewährt. Diese Daten liegen dann auf dem Ressource Server
- Ressource Server: Auf ihm liegen die Ressourcen. Auf Basis von Access Tokens kann Zugriff gewährt werden
- Client: Eine dritte Anwendung, die auf die Ressourcen zugreifen will. Dieser Client wird auf verschiedenen Endgeräten ausgeführt
- Authorization Server: Stellt die Access Tokens für den vom Ressource Owner erlaubten Anwendungsbereich aus
- Man kann diese Methode mit z.B. Facebook und Twitter nutzen

# Quellen

- <http://www.passportjs.org/>
- <https://www.ctl.io/developers/blog/post/build-user-authentication-with-node-js-express-passport-and-mongodb>
- <https://stormpath.com/blog/build-app-nodejs-express-passport-stormpath>