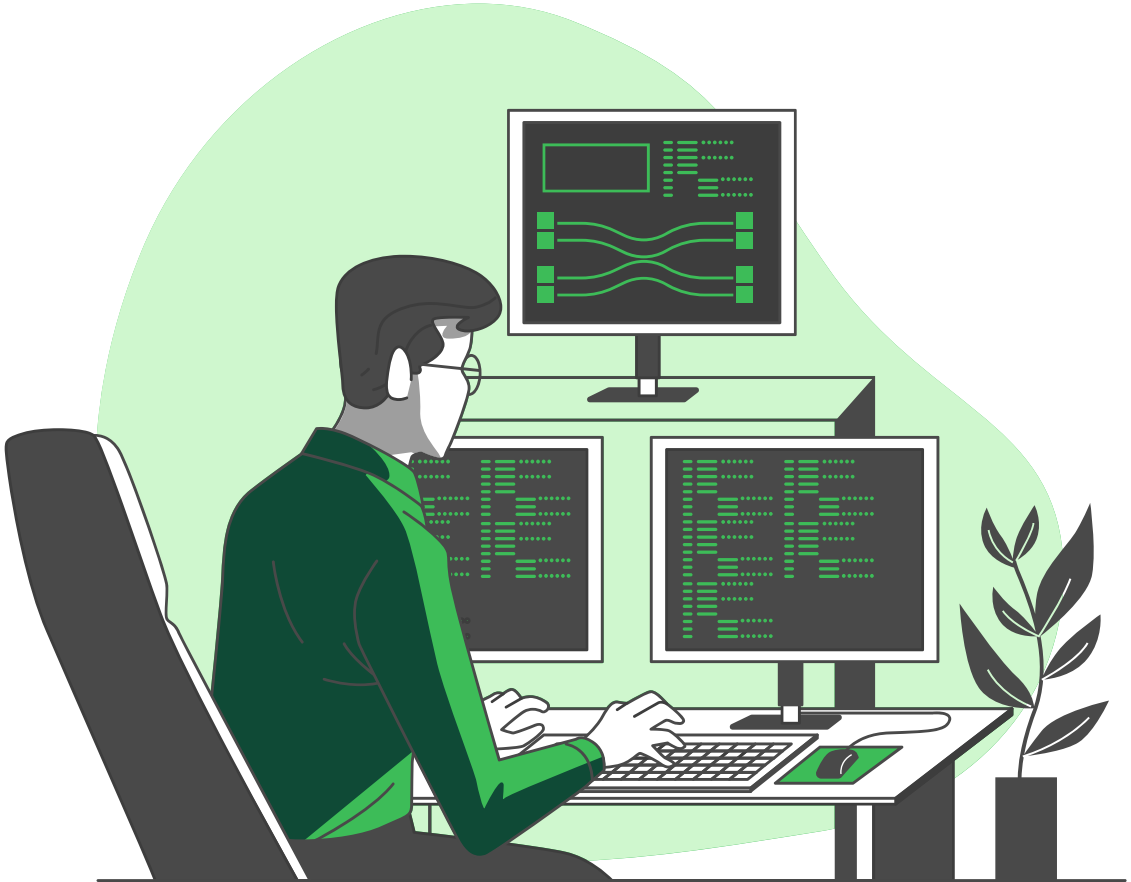


Noteify

Secure Software Engineering

Wer sind wir?

Marcel Kaiser

Informatik
4. Semester

Benjamin Wirth

Informatik
4. Semester

Jonathan Rech

Informatik
4. Semester

Ablauf

01

Aufbau

Genereller Aufbau der Webapp

02

Funktionen

Funktionen der Anwendung, Dependencies, Umsetzung und Sicherheit

03

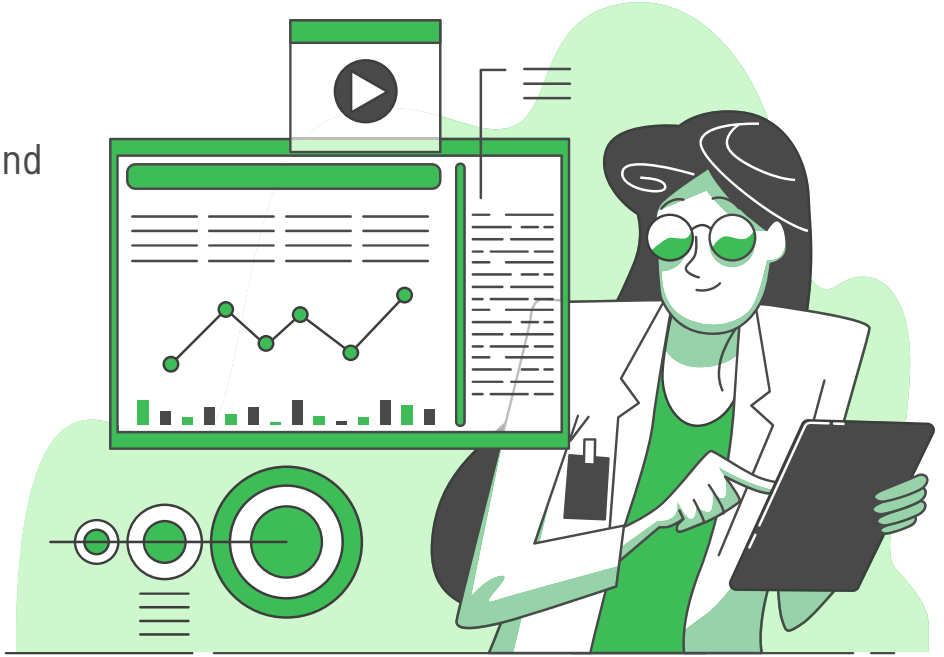
Risiken

Mögliche Sicherheitsrisiken und der Schutz vor diesen

04

Ausblick

Was hätte noch, wenn mehr Zeit und Geld?



01
Aufbau

Genereller Aufbau der Webapp

Aufbau

01 Angular

Frontend

02 Express

Backend

03 MariaDB

Datenbank

04 Docker

Container-Dienst

05 Nginx

Webserver

Angular

Eigenschaften

Frontend-Framework

Typescript

Modular
Wartbar
} Komponentenbasiert

Open-Source

Security

Sanitizer

Viele Anwender

Google (Support gegeben)

Express.js

Eigenschaften

Serverseitiges Webframework

Node

Javascript

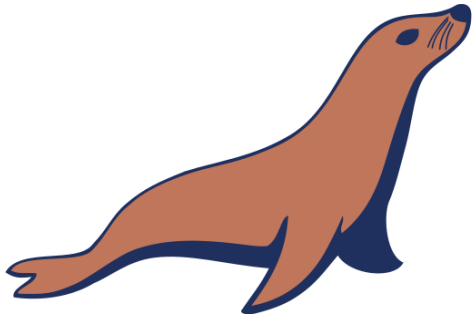
Lightweight

Open-Source

Security

Viele Anwender

Teil der Openjs-Foundation



MariaDB

MariaDB

Eigenschaften

Datenbank

Abspaltung von MySQL

Spektakulär unspektakulär

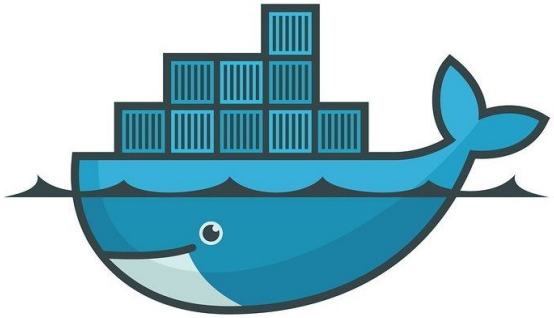
Open Source

Security

Viele Anwender

Großer Entwickler

State Of The Art



Docker

Eigenschaften

Container-Dienst

Plattformübergreifende Nutzbarkeit der Webapp

Open Source

Security

Viele Anwender

Virtuelle Umgebung

Am Markt etabliert

Nginx

Eigenschaften

Webserver-Software

Open Source

Security

Detaillierte CVE-Liste

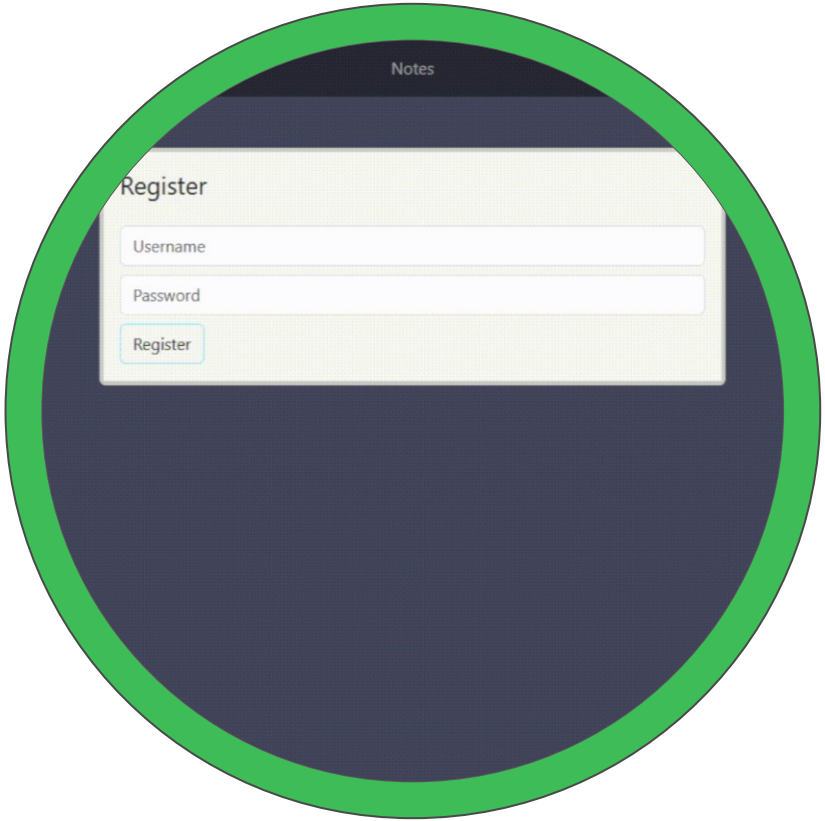
DOS-Protection

Viele weitere (Security-)Features

02
Funktionen

Funktionen der Anwendung, Dependencies, Umsetzung und Sicherheit

Registrierung neuer Benutzer



Benutzername bereits vorhanden?

Passwort Überprüfung

Verschlüsseltes Abspeichern

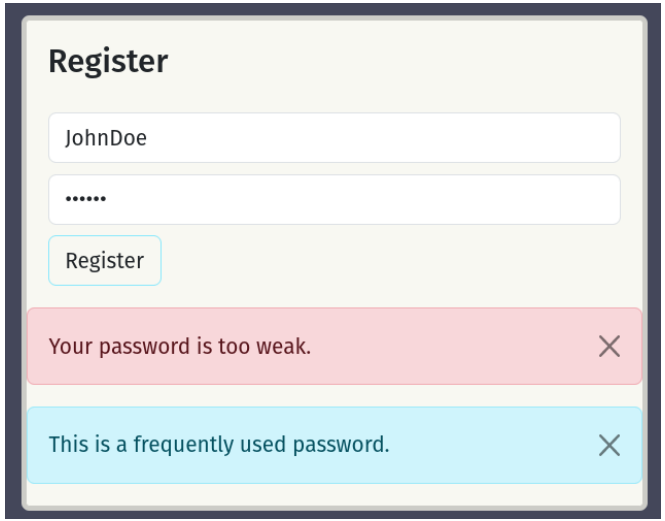
SQL-Injection

Benutzername schon vorhanden

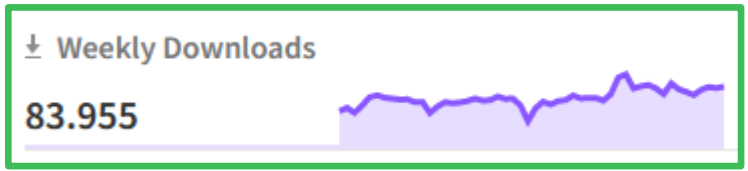
```
async function checkIfUserExists(username){
  const checkUsername = 'SELECT COUNT(*) AS count FROM users WHERE username = ?';
            'SELECT COUNT(*) AS count FROM users WHERE username = ?';
  try{
    const conn = await pool.getConnection();
    const result = await conn.query(checkUsername, [username]);
    conn.release();

    const count = result[0].count;
    return count > 0;
  } catch(error) {
    console.error(error);
    throw error;
  }
}
```

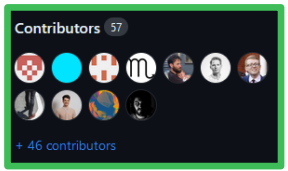
Passwort Überprüfung

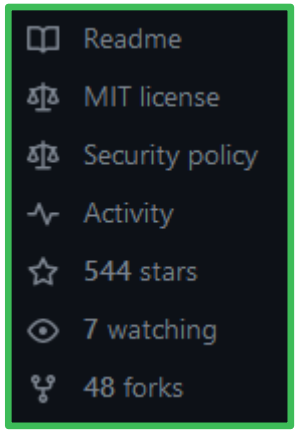
```
const {username, password} = req.body;
const zxcvbnResult = zxcvbn(password, [username]);
const score = zxcvbnResult.score;
const feedback = zxcvbnResult.feedback;
```

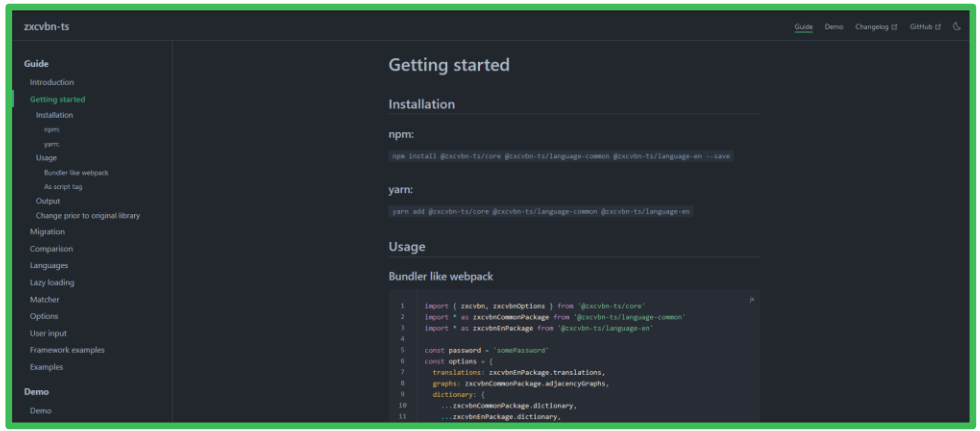


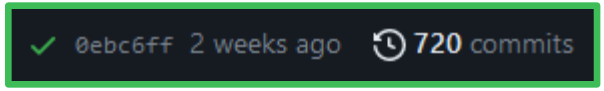
zxcvbn-ts/core









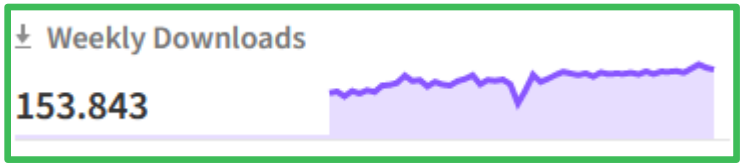


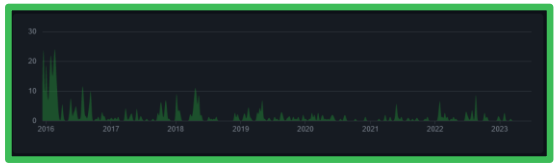
Verschlüsseltes Abspeichern

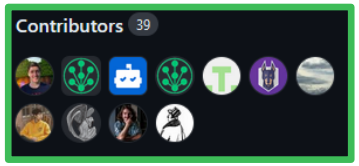
```
const hashed_password = await argon.hash(password.toString());
```

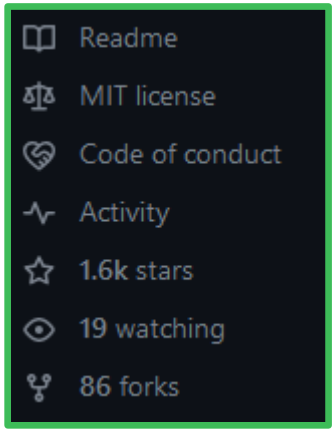
```
async function registerNewUser(username, password){
  const newUser ='INSERT INTO users (username, pass) VALUES (?, ?)';

  try{
    const conn = await pool.getConnection();
    const result = await conn.query(newUser, [username, password]);
    conn.release();

    return 1;
  } catch(error) {
    return 0;
  }
}
```

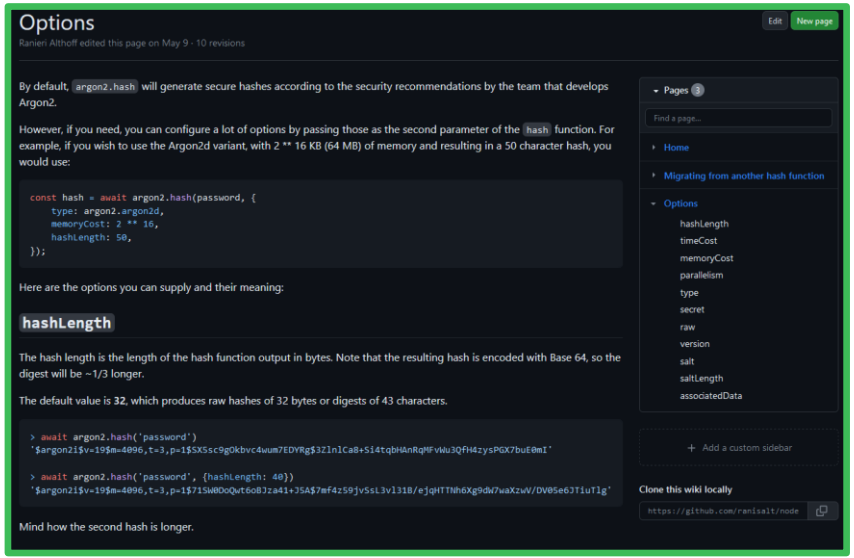
argon2


Weekly Downloads
153.843

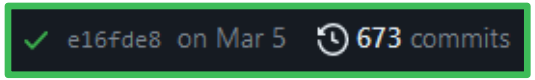

30
20
10
0
2016
2017
2018
2019
2020
2021
2022
2023


Contributors 39


Readme
MIT license
Code of conduct
Activity
1.6k stars
19 watching
86 forks


Options
Ranieri Althoff edited this page on May 9 · 10 revisions
Edit
New page
By default, argon2.hash will generate secure hashes according to the security recommendations by the team that develops Argon2.
However, if you need, you can configure a lot of options by passing those as the second parameter of the hash function. For example, if you wish to use the Argon2d variant, with 2 ** 16 KB (64 MB) of memory and resulting in a 50 character hash, you would use:
const hash = await argon2.hash(password, {
type: argon2.argon2d,
memoryCost: 2 ** 16,
hashLength: 50,
});
Here are the options you can supply and their meaning:
hashLength
The hash length is the length of the hash function output in bytes. Note that the resulting hash is encoded with Base 64, so the digest will be ~1/3 longer.
The default value is 32, which produces raw hashes of 32 bytes or digests of 43 characters.
> await argon2.hash('password')
'$argon2i$v=19$m=4096,t=3,p=1$SX5sc9gOkbvc4wum7EDYRg$3ZlnlCa8+Si4tqbHAnRqMFvWu3QfH4zysPGX7buE0mI'
> await argon2.hash('password', {hashLength: 40})
'$argon2i$v=19$m=4096,t=3,p=1$71SW0DoQwt6oBJza41+J5A$7mf4z59jvSsL3v131B/ejqHTTNh6Xg9dW7waXzwV/DV05e6JTiuTlg'
Mind how the second hash is longer.
Pages
Find a page...
Home
Migrating from another hash function
Options
hashLength
timeCost
memoryCost
parallelism
type
secret
raw
version
salt
saltLength
associatedData
Add a custom sidebar
Clone this wiki locally
https://github.com/ranisalt/node


BUNDLE SIZE
330.7 kB
MINIFIED
96.5 kB
MINIFIED + GZIPPED
DOWNLOAD TIME
1.93 s
SLOW 3G
110 ms
EMERGING 4G


e16fde8 on Mar 5
673 commits

Benutzer Login

Stimmen Benutzername und Passwort?

Fehlermeldung uneindeutig

JSON Web Token

SQL-Injection

Passwort Überprüfung

```
const sql = `SELECT * FROM users WHERE username = ?`;
const result = await con.query(sql, [username]);
if (result.length > 0) {
```

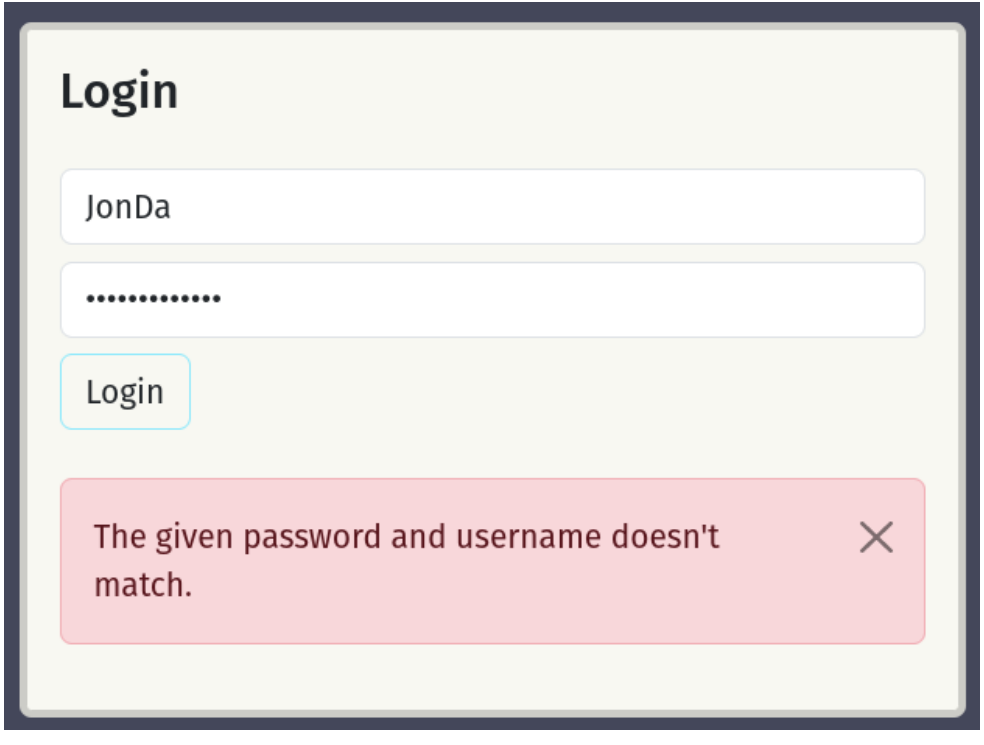
```
const isMatch = await argon2.verify(hashedPassword, password.toString());
```

hashedPassword = Aus Datenbank

password = aus HTML-Form (Frontend)

Uneindeutige Fehlermeldung

```
res.send({status:0, error: 'invalid Username or Password', msg:'invalid Username or Password'});
```



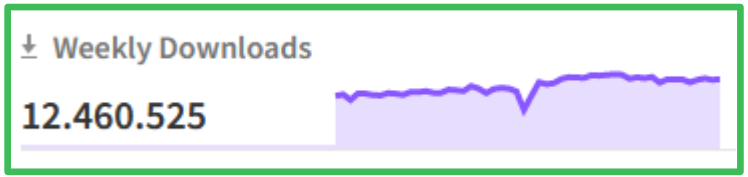
JSON Web Token

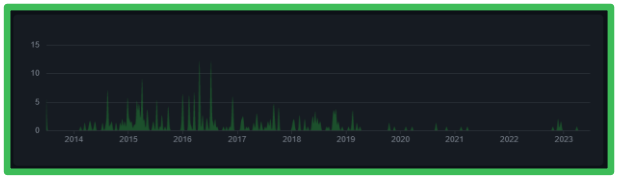
```
let token = jwt.sign({username:username, user_id: userIdQuery[0].user_id.toString()},
                                                    jwtSecret,{ expiresIn: '1h' })
```

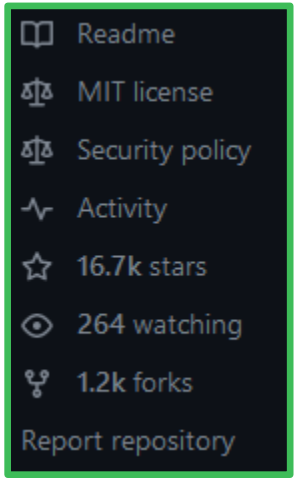
jwtSecret = Enviroment-Variable

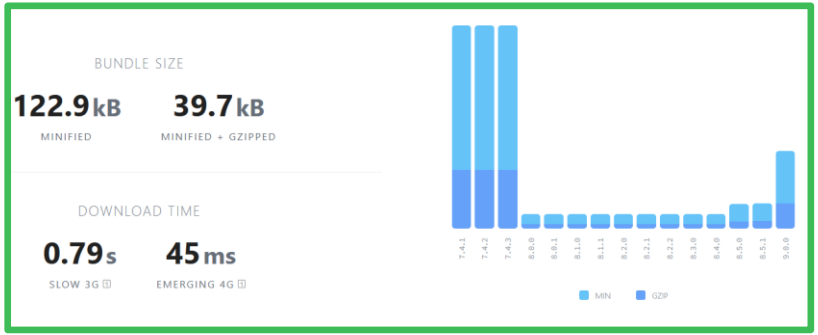
JWT läuft nach einer Stunde ab

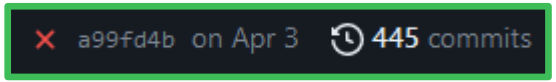
jsonwebtoken











Notiz anlegen

Benutzerüberprüfung

Spam-Protection

SQL-Injection

Benutzerüberprüfung

```
router.post('/new',authenticateToken, async function (req, res) {
```

```
authenticateToken,
```

```
jwt.verify(req.headers.authorization, jwtSecret, function (err, decoded) {
    if(decoded){
        req.user = decoded.user_id;
        userID = decoded.user_id;
        if(!search){next();}
    }else{
      if(err.name === 'TokenExpiredError') res.send({ message: "Token expired" });
      else  {res.status(401).send({ message: "Unauthorized" }); isAuthenticated = false;}
```

Notiz anlegen

```
const query = "INSERT INTO notes(note_id,titel, isPrivate,content, user_id, created, lastChanged, y VALUES (?,?,?,?,?,?,?,?)"
conn = await pool.getConnection();
const date = new Date().toISOString().slice(0, 19);
const note = await conn.query(quer el,isPrivate, content, authorId,date,date,youtube]);
                              [uuidv4(),
```

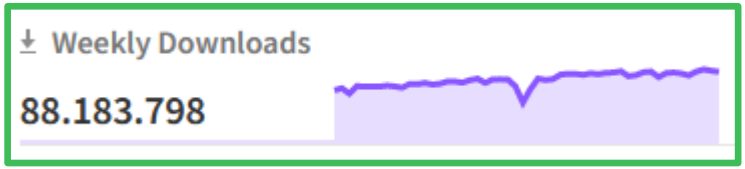
uuidv4 ist garantiert random

State-Of-The-Art

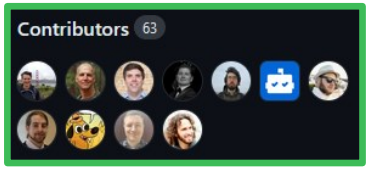
Einmalig für:

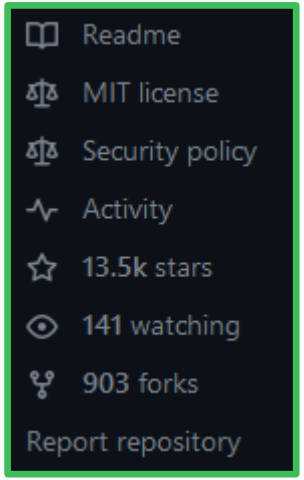
- Gerät
- Uhrzeit
- 128-Bit zufällige Zahl

uuid


Weekly Downloads
88.183.798

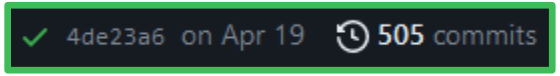

60
40
20
0
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023


Contributors 63

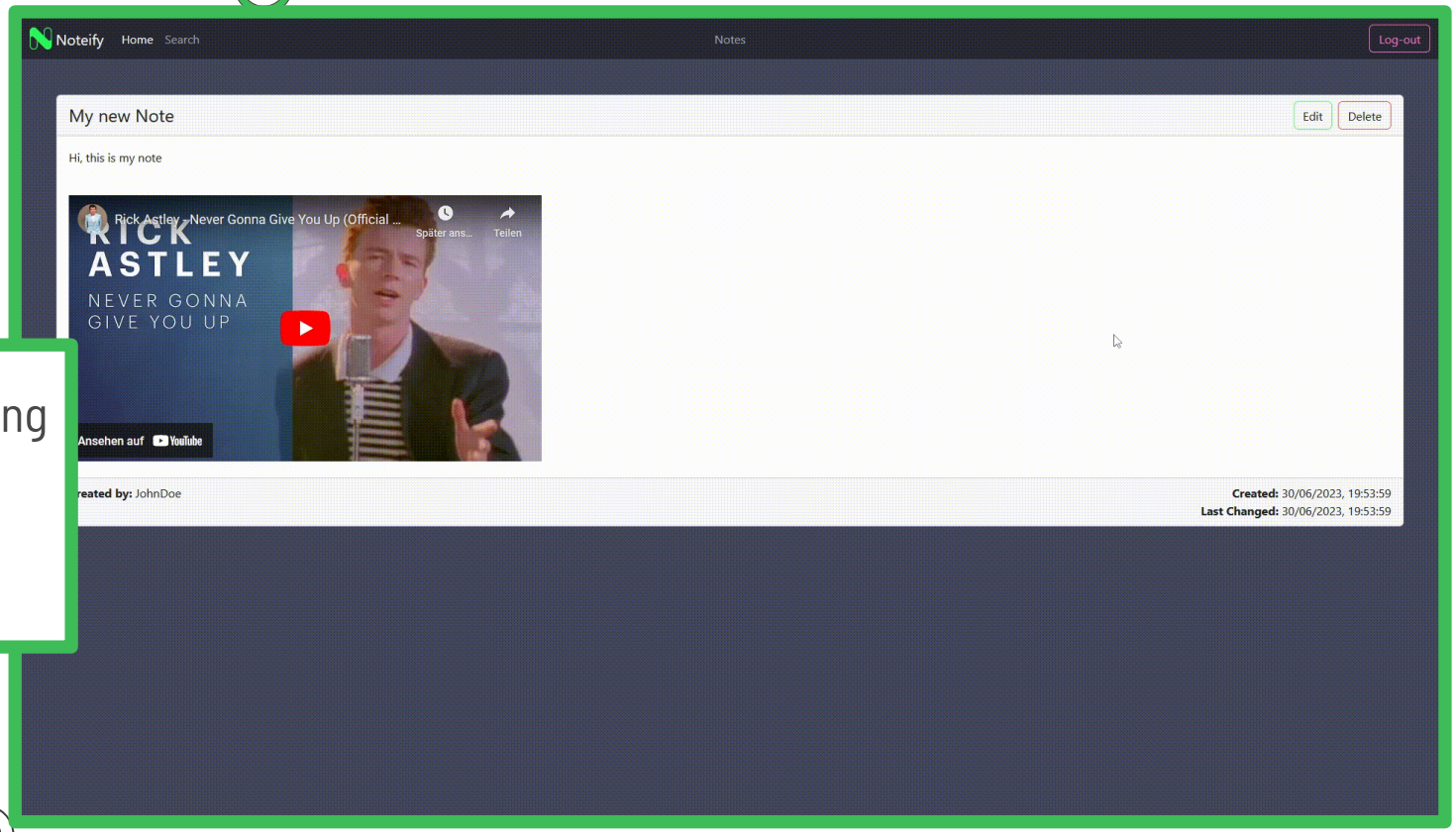

Readme
MIT license
Security policy
Activity
13.5k stars
141 watching
903 forks
Report repository


README.md
uuid
CI passing
Browser passing
For the creation of RFC4122 UUIDs
Complete - Support for RFC4122 version 1, 3, 4, and 5 UUIDs
Cross-platform - Support for ...
CommonJS, ECMAScript Modules and CDN builds
Node 12, 14, 16, 18
Chrome, Safari, Firefox, Edge browsers
Webpack and rollup.js module bundlers
React Native / Expo
Secure - Cryptographically-strong random values
Small - Zero-dependency, small footprint, plays nice with "tree shaking" packagers
CLI - Includes the uuid command line utility
Note Upgrading from uuid@3? Your code is probably okay, but check out Upgrading From uuid@3 for details.
Note Only interested in creating a version 4 UUID? You might be able to use crypto.randomUUID(), eliminating the need to install this library.
Quickstart
To create a random UUID...
1. Install
npm install uuid
2. Create a UUID (ES6 module syntax)


BUNDLE SIZE
8.3 kB
MINIFIED
3.3 kB
MINIFIED + GZIPPED
DOWNLOAD TIME
66 ms
SLOW 3G
4 ms
EMERGING 4G
MIN
GZIP


4de23a6 on Apr 19
505 commits

Notiz editieren



- Benutzerüberprüfung
- Spam-Protection
- SQL-Injection

Benutzerüberprüfung

```
router.post('/update/:id',authenticateToken, async function (req, res) {
                          authenticateToken,
```

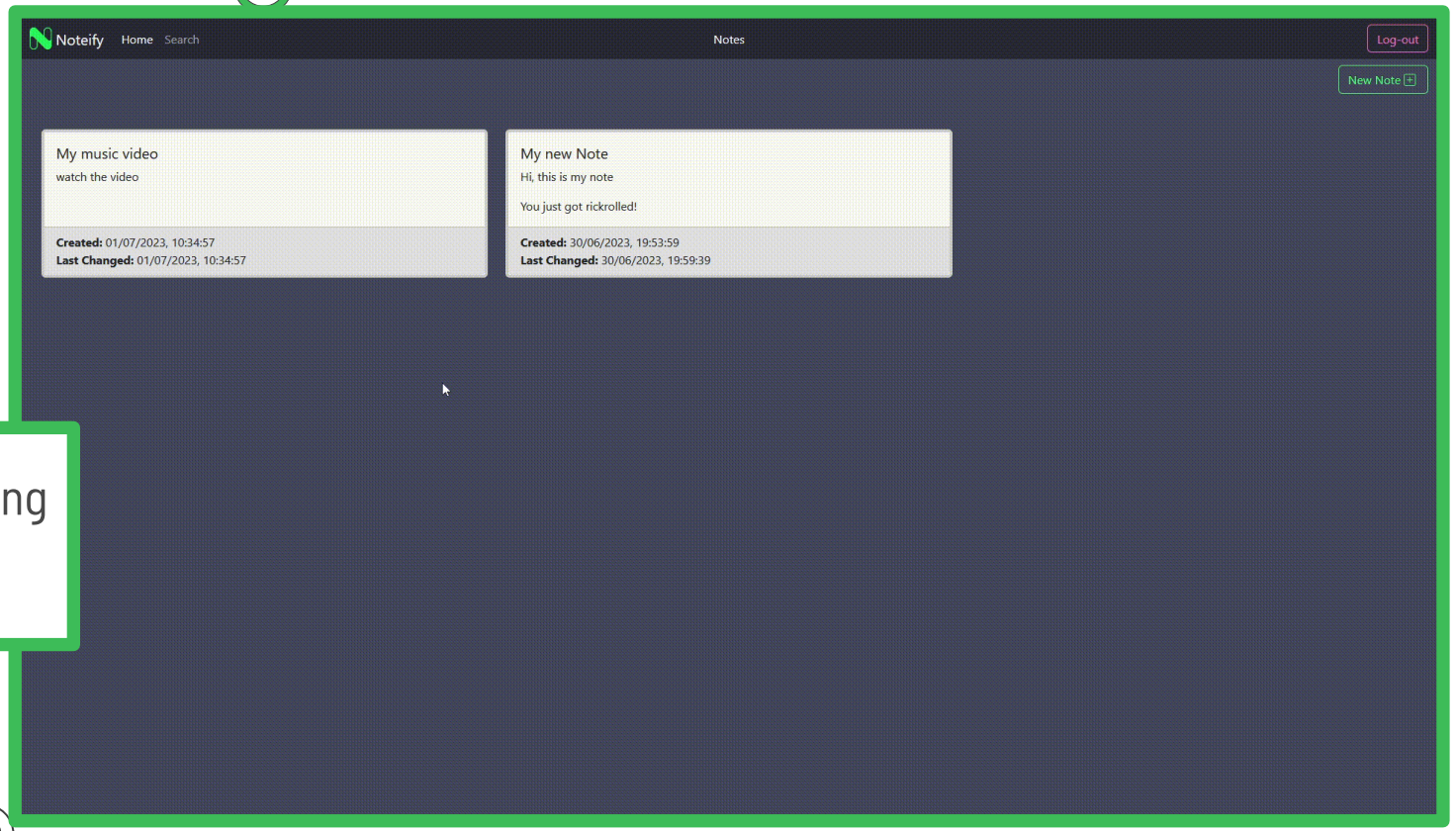
```
const query2 = "SELECT user_id FROM notes WHERE note_id = ?";
authorIdQuery = await conn.query(query2,[id]);
if(authorId == authorIdQuery[0].user_id){
```

Notiz editieren

```
"UPDATE notes SET titel = ?, isPrivate = ?, content = ?,  lastChanged = ?, youtube = ? WHERE notes.note_id = ?"
```

```
const note = await conn.query(query, [titel,isPrivate, content,date,youtube, id]);
```

Notiz löschen



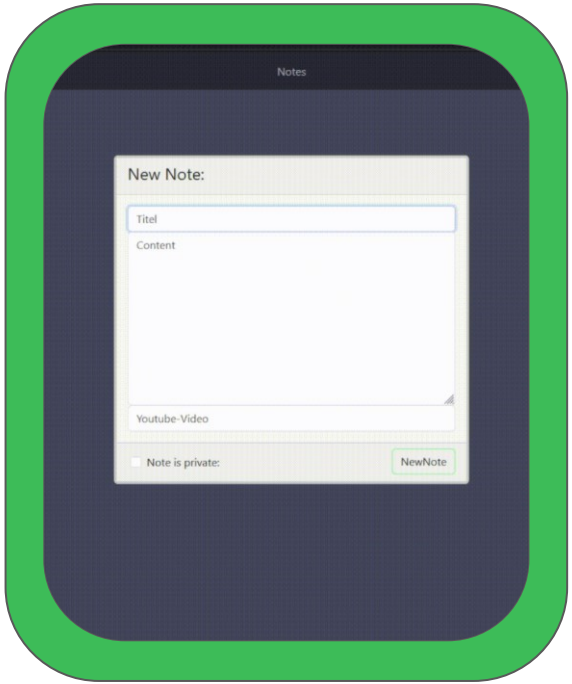
- Benutzerüberprüfung
- SQL-Injection

Benutzerüberprüfung + Notiz löschen

```
router.delete('/singlenote/:id', authenticateToken, async function (req, res) {
                                 authenticateToken,
```

```
const query = "DELETE FROM notes WHERE notes.note_id = ?"
conn = await pool.getConnection();
const query2 = "SELECT user_id FROM notes WHERE note_id = ?";
authorIdQuery = await conn.query(query2,[id]);
if(authorId == authorIdQuery[0].user_id){
  const note = await conn.query(query, [id]);
```

Youtube-Video an Notiz anhängen



Schadcode abfangen

URL-Überprüfung

Regex

```
var regExp = /^.*((youtu.be\/)|(v\/)|(\/u\/\w\/)|(embed\/)|(watch\?))\??v?=?([^#&?]*).*/;
```

https://m.youtube.com/watch?v=**lalOy8Mbfdc**
https://youtu.be/**oTJRivZTMLs**&feature=channel
http://m.youtube.com/v/**0zM3nApSvMg**?fs=1&hl=en_US&rel=0
http://youtu.be/-**wtIMTCHWuI**

Darstellung des iFrames

```
<youtube-player videoId="{{notes[0].youtube}}" suggestedQuality="highres" [height]="360" [width]="640"></youtube-player>
```

<youtube-player

Eigene Funktion von Angular

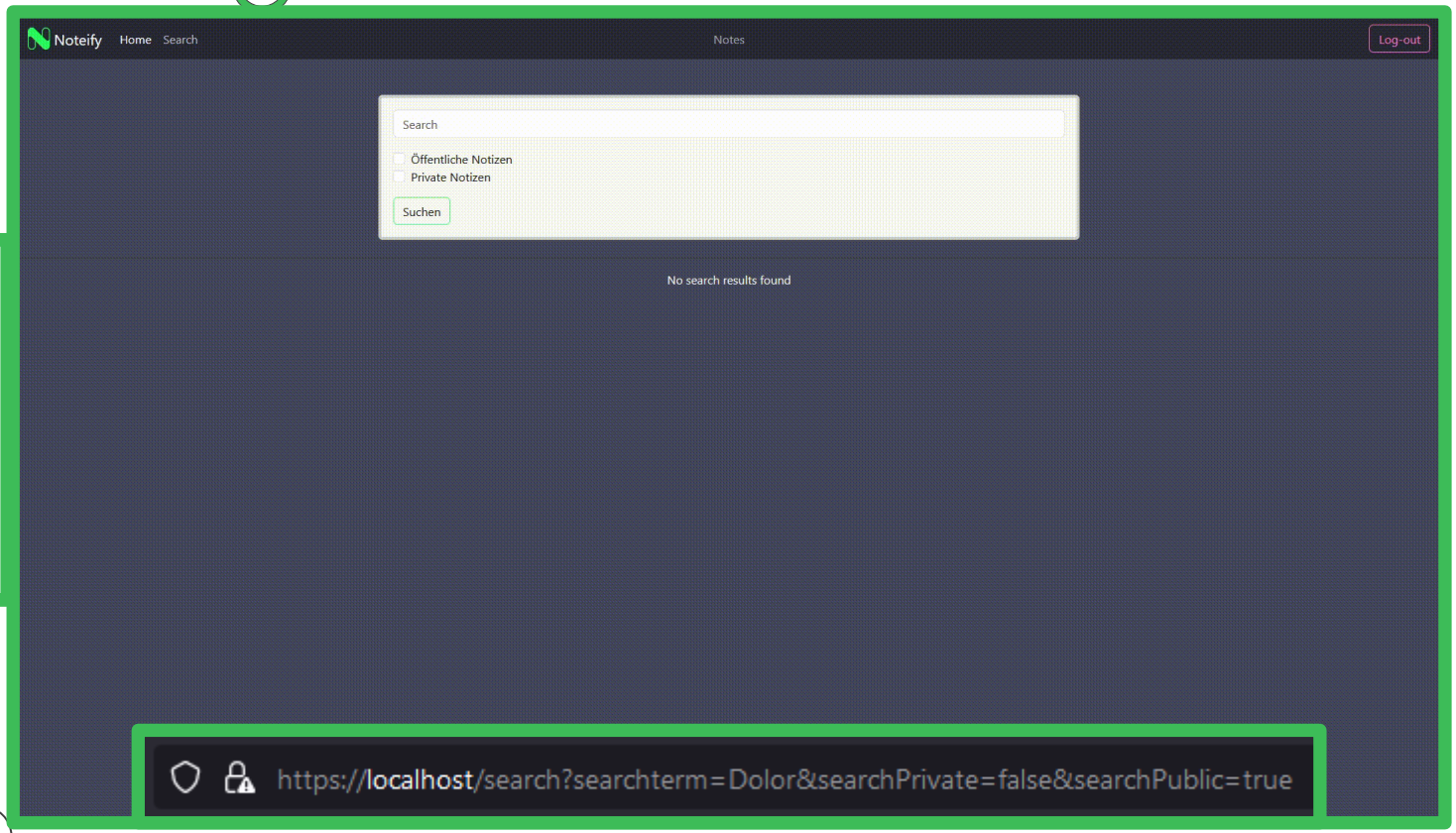
Keine Darstellung, wenn fehlerhafte URLs oder Schadcode

Suchen von Notizen

XSS-Protection

SQL-Injection

Authentifizierung



XSS-Protection

```
<p class="text-muted mt-3" *ngIf="searchterm">Suchbegriff: {{ searchterm }}</p>
```

Eigene Funktion von Angular

Angular übernimmt Sanitization

Suchen von Notizen

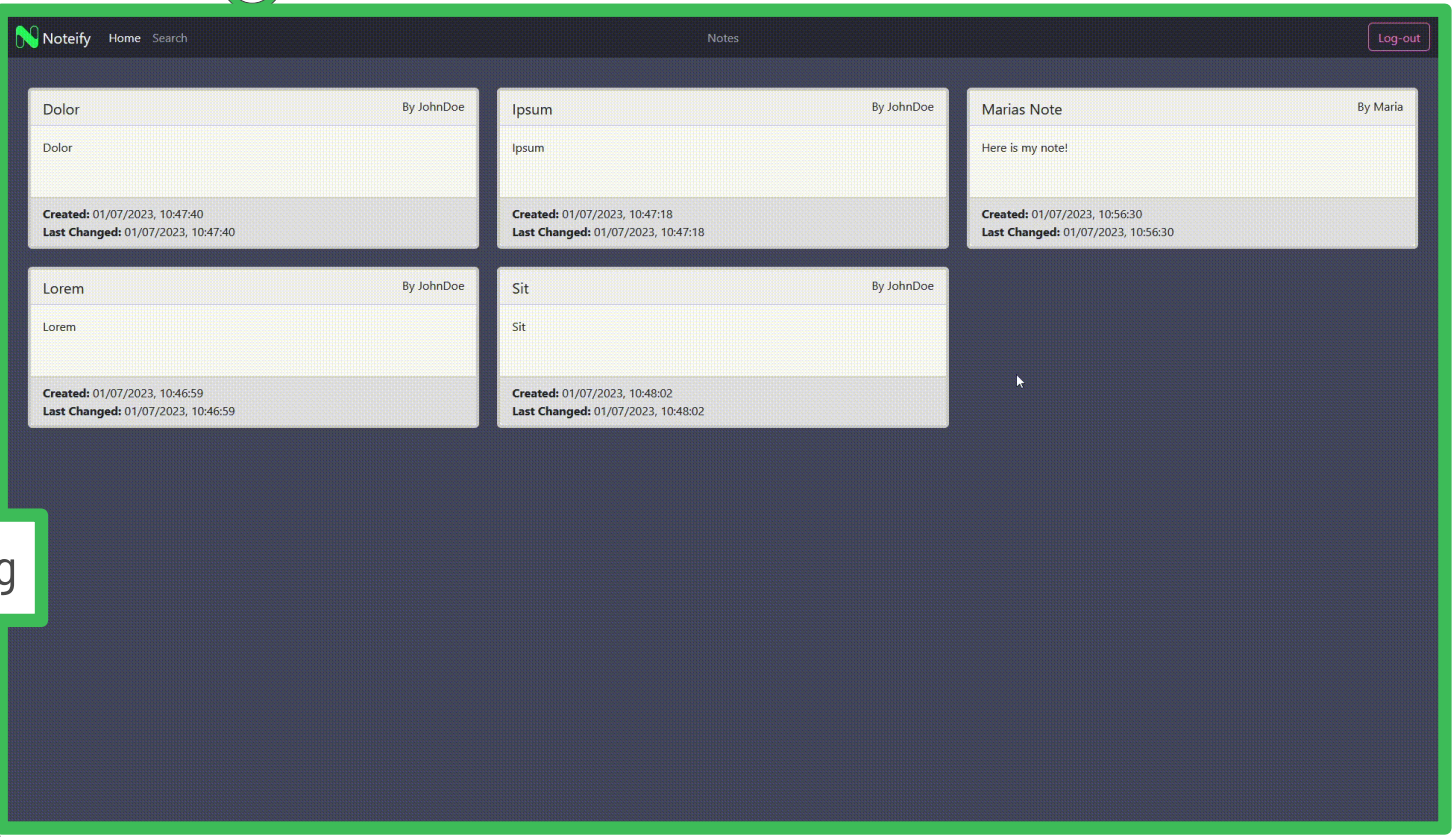
```
const query ="SELECT notes.* from notes JOIN users ON notes.user_id = \
              users.user_id WHERE ((notes.titel LIKE CONCAT('%', ?, '%') OR \
              notes.content LIKE CONCAT('%', ?, '%') OR \
              users.username LIKE CONCAT('%', ?, '%')) \
              AND notes.isPrivate = 1) AND users.user_id = ?";
```

Private Search →

```
if (searchPrivate) {
    authenticateToken(req,res,next,true);
```

Prepared Statement ist äquivalent in „public Search“

Anzeigen von öffentlichen/eigenen Notizen



Benutzerüberprüfung

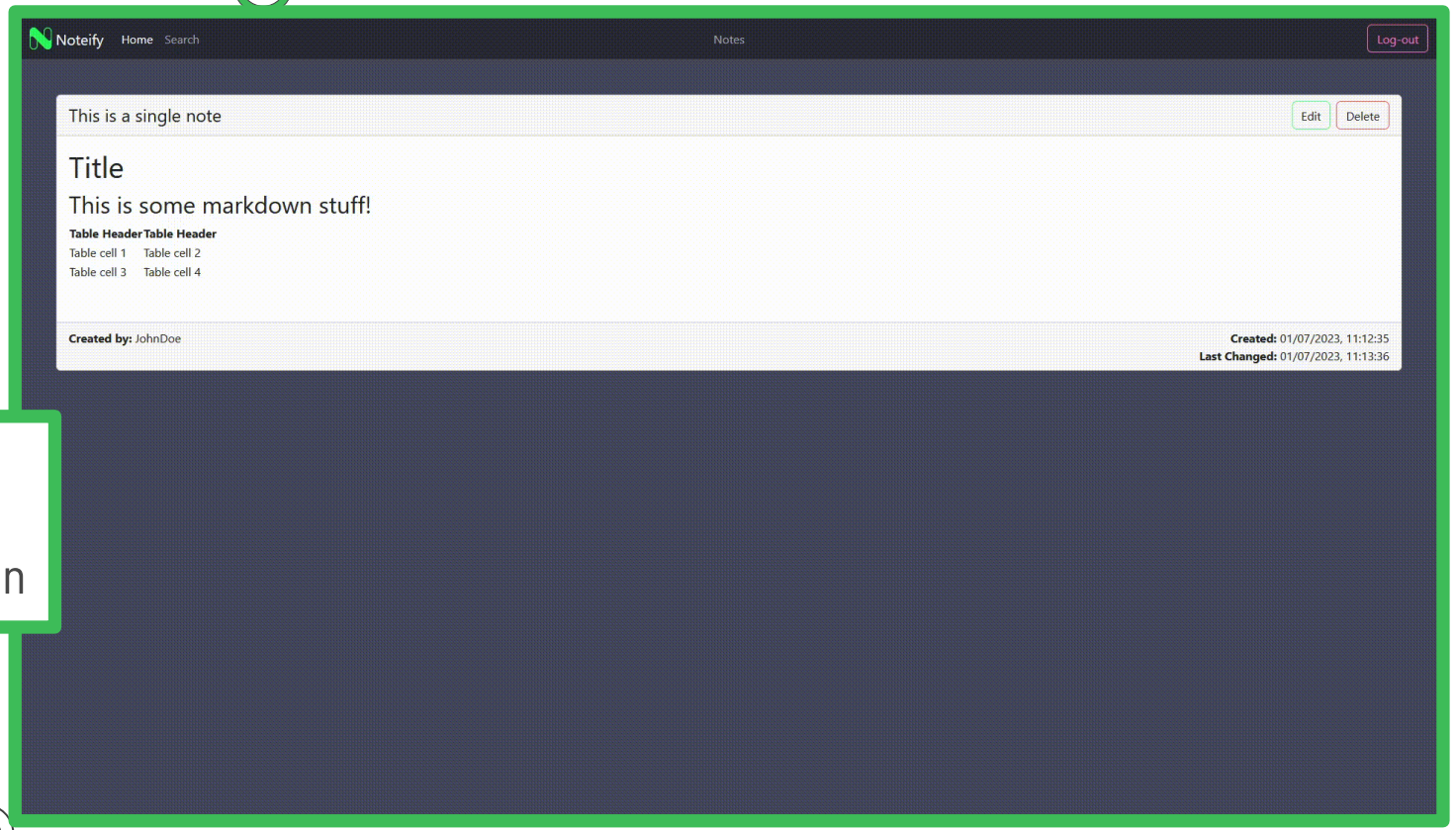
Benutzerüberprüfung + eigene Notizen

```
router.get('/usernotes',authenticateToken, async function (req, res) {
                        authenticateToken,
```

```
const query = "SELECT notes.*,users.username FROM notes JOIN users \
                ON notes.user_id = users.user_id WHERE notes.user_id = ?";
```

Öffentliche Notizen: Ohne Benutzerüberprüfung und andere Where-Bedingung

Anzeigen einzelner Notizen



XSS-Protection

Private Notizen lesen

XSS-Protection

```
<p class="card-text" [innerHTML]="notes[0].content"></p>
```

Eigene Funktion von Angular

Angular übernimmt Sanitization

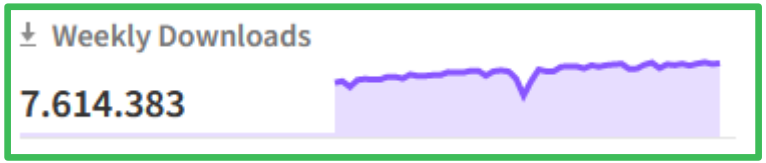
Markdown- und HTML-Support

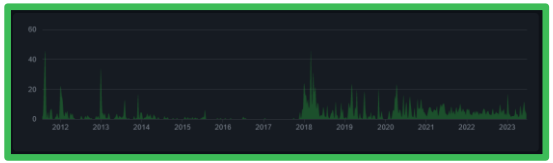
```
elem.content = marked.marked.parse(elem.content.replace(/^[\u200B\u200C\u200D\u200E\u200F\uFEFF]/,""))
```

marked.marked.parse

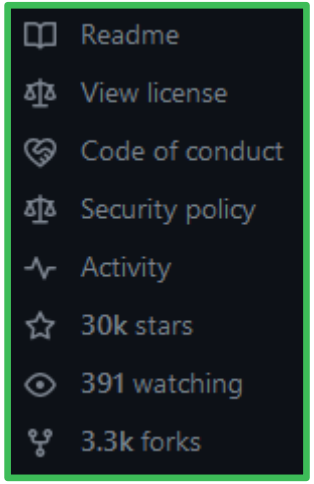
HTML-Support durch [innerHTML]-Tag gegeben

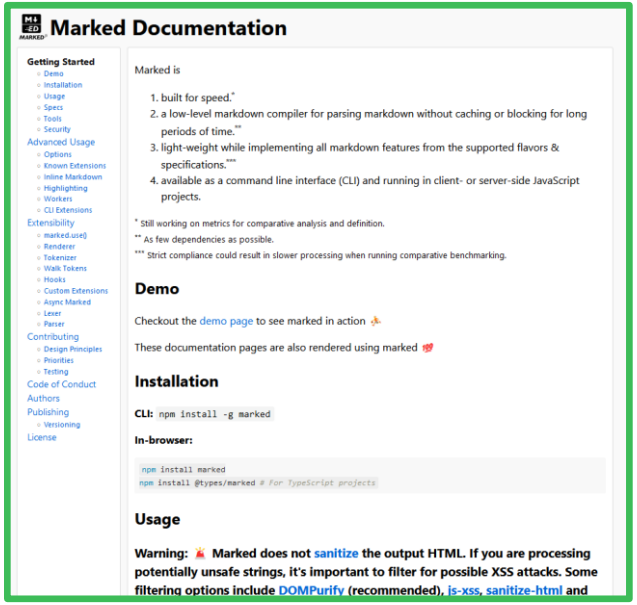
marked

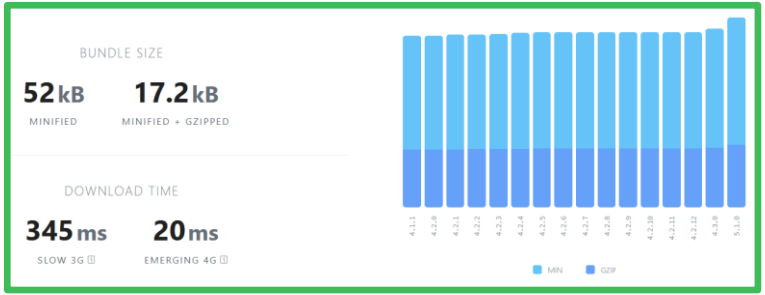

Weekly Downloads
7.614.383

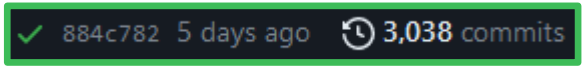

60
40
20
0
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023


Contributors 161

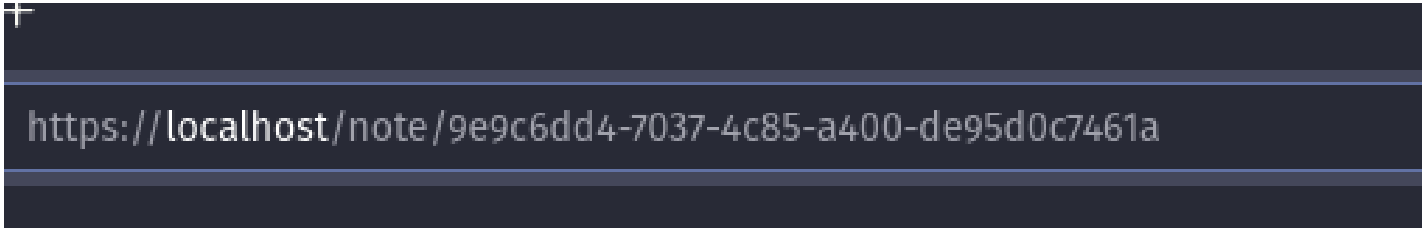

Readme
View license
Code of conduct
Security policy
Activity
30k stars
391 watching
3.3k forks


Marked Documentation
Getting Started
Demo
Installation
Usage
Specs
Tools
Security
Advanced Usage
Options
Known Extensions
Inline Markdown
Highlighting
Workers
CLI Extensions
Extensibility
marked.use()
Renderer
Tokenizer
Walk Tokens
Hooks
Custom Extensions
Async Marked
Lexer
Parser
Contributing
Design Principles
Priorities
Testing
Code of Conduct
Authors
Publishing
Versioning
License
Marked is
1. built for speed.*
2. a low-level markdown compiler for parsing markdown without caching or blocking for long periods of time.**
3. light-weight while implementing all markdown features from the supported flavors & specifications.***
4. available as a command line interface (CLI) and running in client- or server-side JavaScript projects.
* Still working on metrics for comparative analysis and definition.
** As few dependencies as possible.
*** Strict compliance could result in slower processing when running comparative benchmarking.
Demo
Checkout the demo page to see marked in action
These documentation pages are also rendered using marked
Installation
CLI: npm install -g marked
In-browser:
npm install marked
npm install @types/marked # for TypeScript projects
Usage
Warning: Marked does not sanitize the output HTML. If you are processing potentially unsafe strings, it's important to filter for possible XSS attacks. Some filtering options include DOMPurify (recommended), js-xss, sanitize-html and


BUNDLE SIZE
52 kB
MINIFIED
17.2 kB
MINIFIED + GZIPPED
DOWNLOAD TIME
345 ms
SLOW 3G
20 ms
EMERGING 4G
MIN
GZIP


884c782 5 days ago
3,038 commits

Private Notizen lesen



UUIDv4:

- Kaum Guessable
- Bruteforce kaum möglich

03
Risiken

Mögliche Sicherheitsrisiken und der Schutz vor diesen

Unsichere Kommunikation

- HTTPS zwischen Frontend und Backend
- Erzwingen von HTTPS-Verbindungen des Clients

XSS Angriffe

- Angular Sanitizer

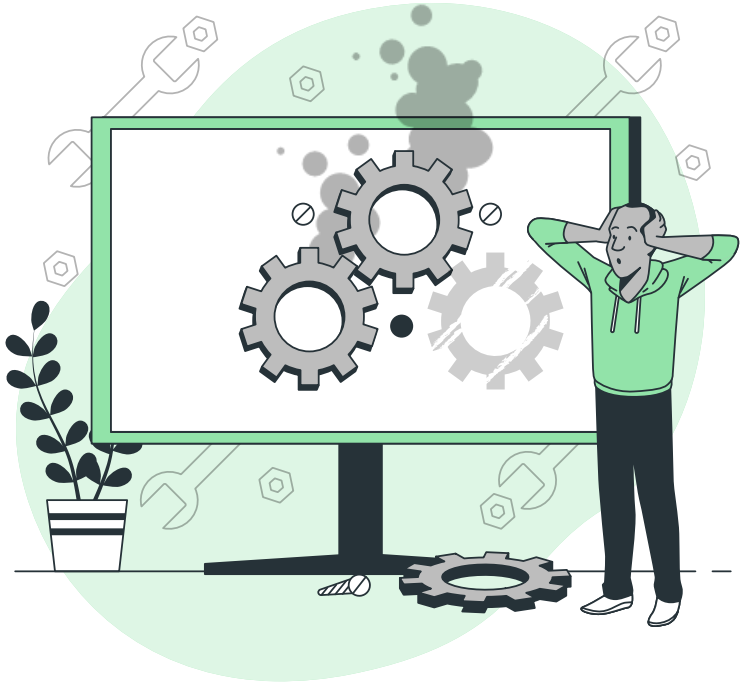
Denial of Service & Brute-Force

Nginx-Konfiguration

- Rate-Limitting

Express-Konfiguration

- Rate-Limitting

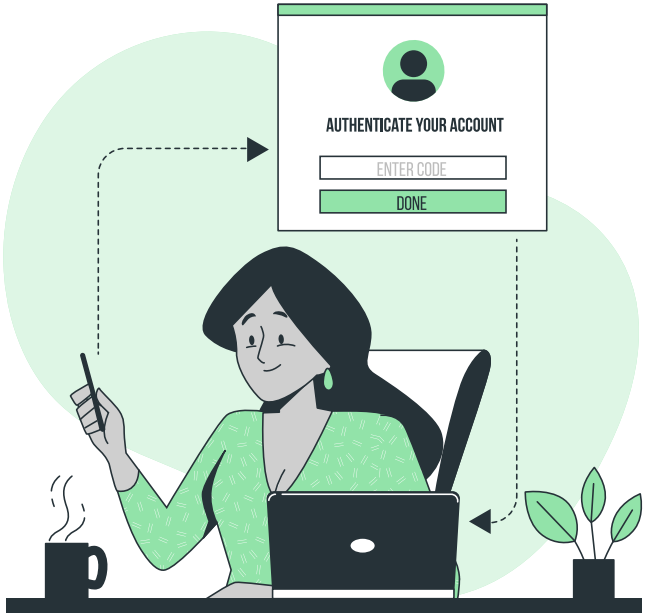


Passwörter

- ZXCVBN: Enforcen starker Passwörter

Sessions

- JSON Web Tokens
- Laufzeit < 1 Stunde



Datenbank

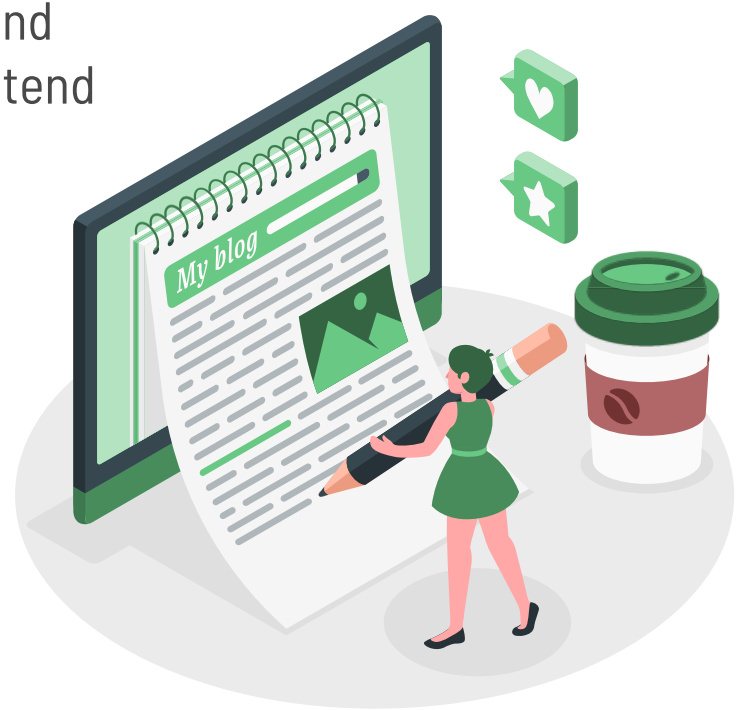
- Datenbank nur vom Container aus erreichbar

CI/CD

- Dependabot
- **Github-Actions**
- CodeQL
- Frontend-Test
- Backend-Test

Security/Logging/Monitoring-Failures

- Überall uneindeutige Fehlermeldungen
- Keine Fehlermeldungen von Backend an Frontend
- Keine Fehlermeldungen von Datenbank an Frontend



04
Ausblick

Was hätte noch, wenn mehr Zeit und Geld?

Was muss noch getan werden?

- Impressum und Datenschutzerklärung
- Web-Application-Firewall
- DMZ einrichten
- Verschlüsseln aller Daten in der Datenbank
- Mehr Pen-Testing (ZAP, Greenbone)
- Echte signierte Zertifikate
- Datenbank Backup Strategie

- Passwort vergessen
- Oauth-Funktionalität

Danke!

Zeit für eine Live-Demo ☺

https://github.com/marcel951/Noteify