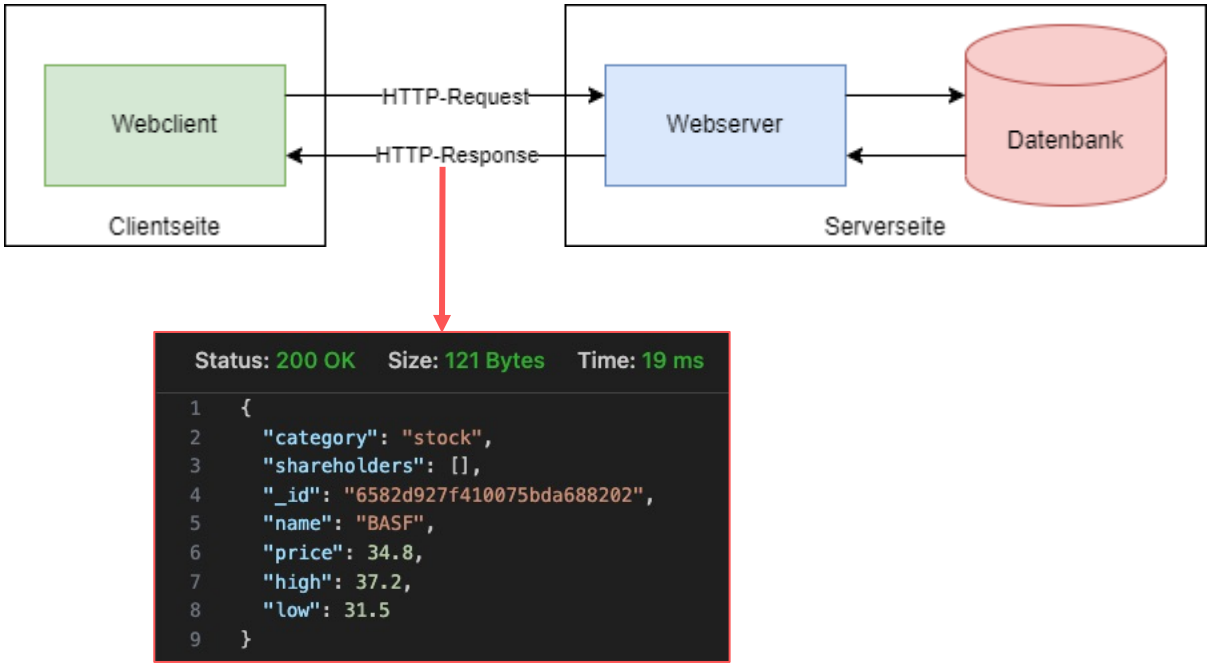
Webtechnologien

Authentifizierung

Stephan Schiffner

Motivation



Problem: Zugriff auf alle Endpoints möglich → alle Daten einsehbar

Authentifizierung

Einführung

Authentifizierung

Aufgaben

- Prüfung Identität eines Benutzers (Person oder System)
- Ist der Benutzer, wer er vorgibt zu sein?

Hauptziel

- Identität bestätigen und Vertrauen sicherstellen

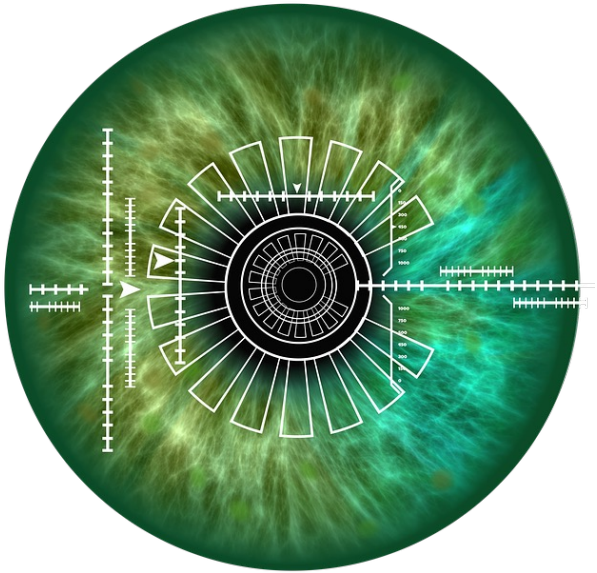
Ergebnis

- Benutzer erhält Zugriff auf Ressourcen und/oder Dienste

Typische Verfahren zur Authentifizierung

Typische Verfahren zur Authentifizierung

- Benutzername und Passwort
- Biometrische Verfahren
- Multi-Faktor Authentifizierung



Abgrenzung Autorisierung

Aufgaben

- Zugriff auf bestimmte Funktionen, Ressourcen, etc.
- Abhängig von der Identität und Attributen

Typische Verfahren

- Rollen, Rechte

Ergebnis

- Benutzer (Person oder System) erhält Zugriff auf Ressourcen und/oder Funktionen

JSON Web Tokens (JWT)

Verfahren zur Authentifizierung

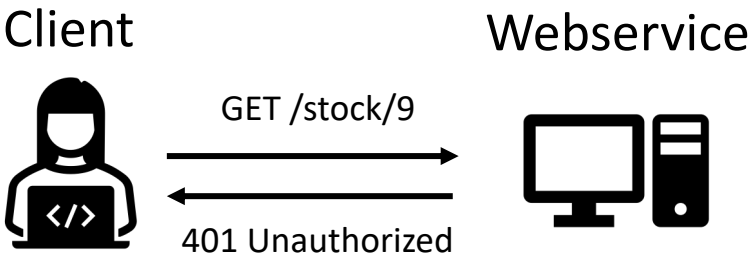
Authentifizierung

Ziel

→ Benutzerkonten mit Benutzername und Passwort anlegen
→ Vor der Abfrage von Endpoints: Anmeldung nötig (also authentifizieren)

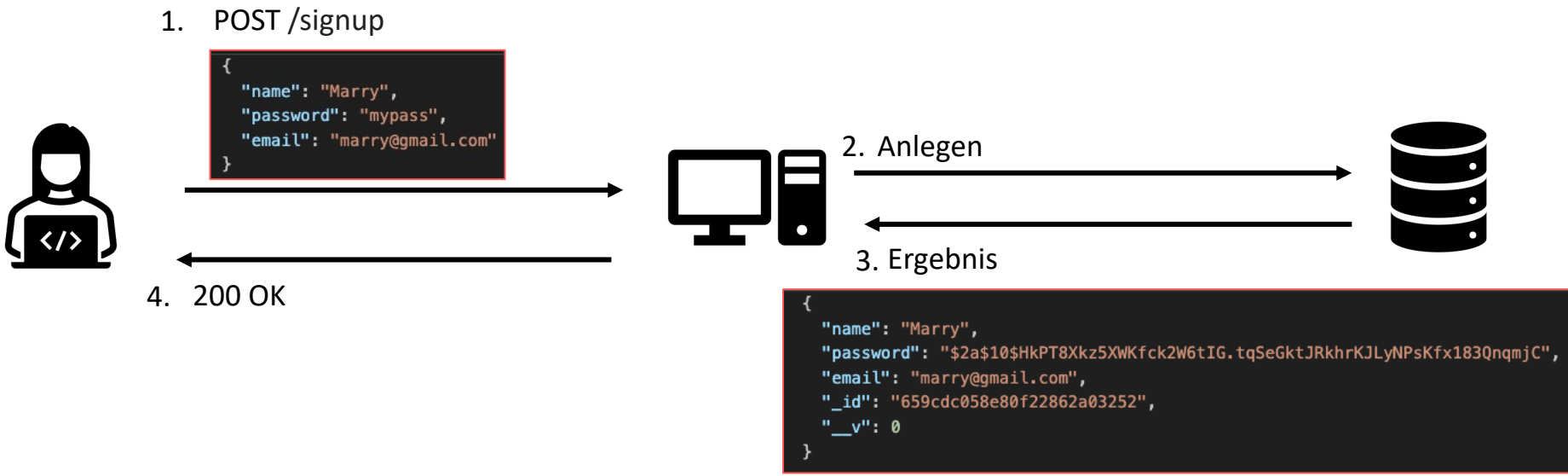
Beispiel

Gewünschtes Verhalten: User hat sich vor dem GET-Request an /stock/:id nicht authentifiziert. Der Webservice lehnt den Request ab.



Schritt 1: Benutzerkonto anlegen

Für jede Person und/oder System wird ein Benutzerkonto angelegt. In diesem Beispiel über einen entsprechenden Endpoint.



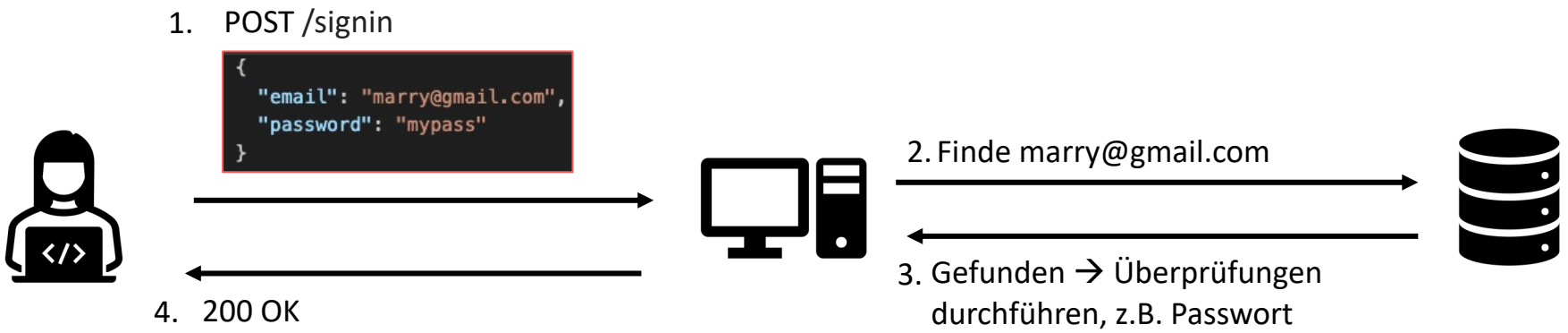
JSON Web Tokens

JSON Web Tokens are an open, industry standard **RFC 7519** method for representing claims securely between two parties.

- Verfahren zur sicheren Übertragung von Informationen in Form von JSON Objekten.
- JWTs werden mit Secret signiert, z.B. mittels HMAC Algorithmus
- Anwendungsfälle: Authentifizierung, sicherer Informationsaustausch

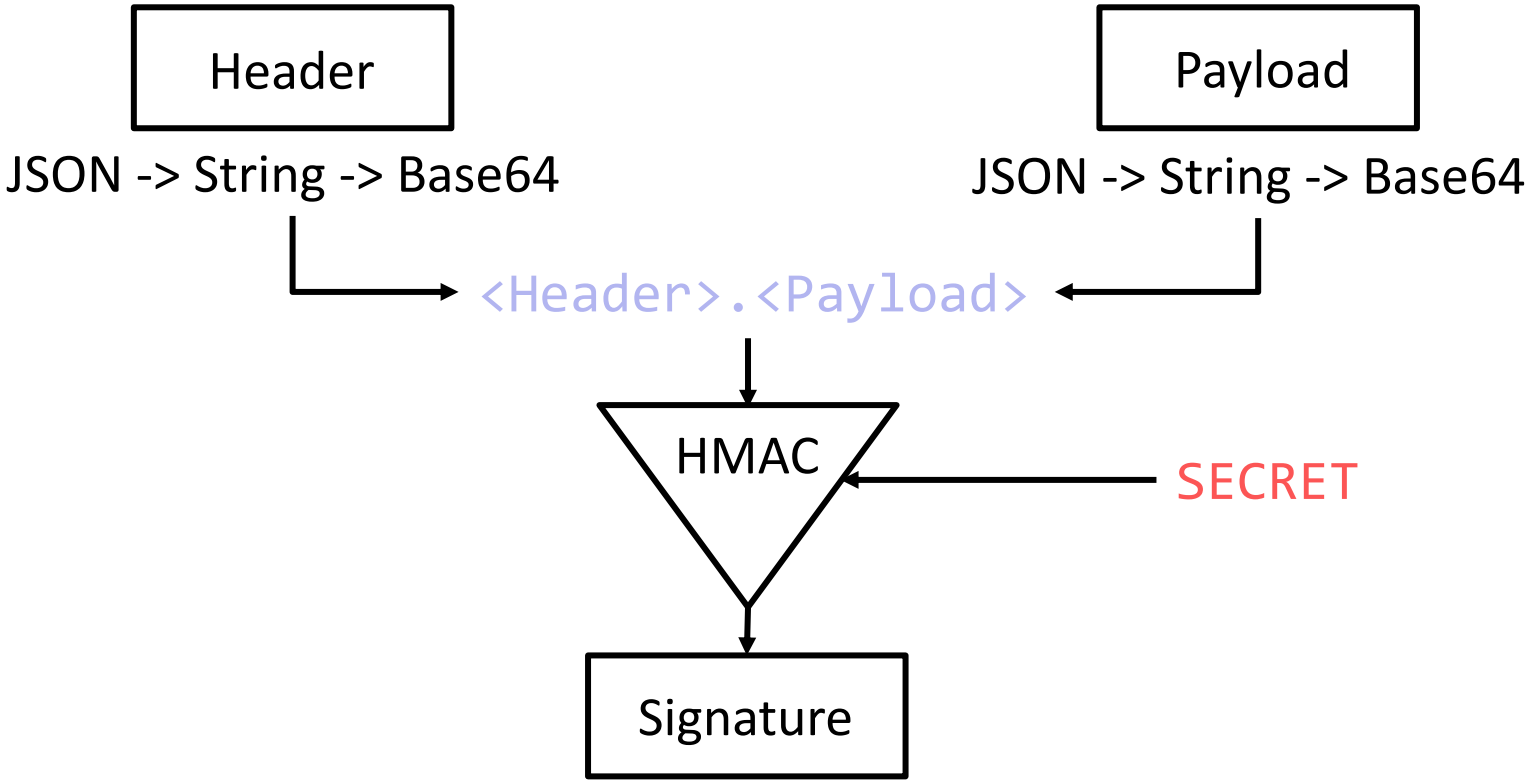
Schritt 2: Anmeldung

Anmeldung mit Username und Passwort. Falls das Tupel aus **Username, Passwort** in der Datenbank gefunden wird und korrekt ist, erhält der User im Gegenzug ein Token.




```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
      .eyJfaWQiOiI2NTljZGMwNThlODBmMjI4NjJhMDMyNTIiLCJpYXQiOjE3MDQ4NjgyNTYsImV4cCI6MTcwNDg3OTA1Nn0
      .DRZLcv8O77n_F2-VLl6tx89JLDTvJFfBG7V60zK3Nh0"
}
```

Erstellung Token



<Header>.<Payload>.<Signature>

JWT Token (Debugger)

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2NTljZGMwNThlODBmMjI4NjJhMDMyNTIiLCJpYXQiOjE3MDQ4NjgyNTYsImV4cCI6MTcwNDg3OTA1Nn0.zQfLCEykAJhH7gp6gtBl3X5Js-JW-TllQ-cfJXQ1W2I
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "_id": "659cdc058e80f22862a03252",
  "iat": 1704868256,
  "exp": 1704879056
}
```

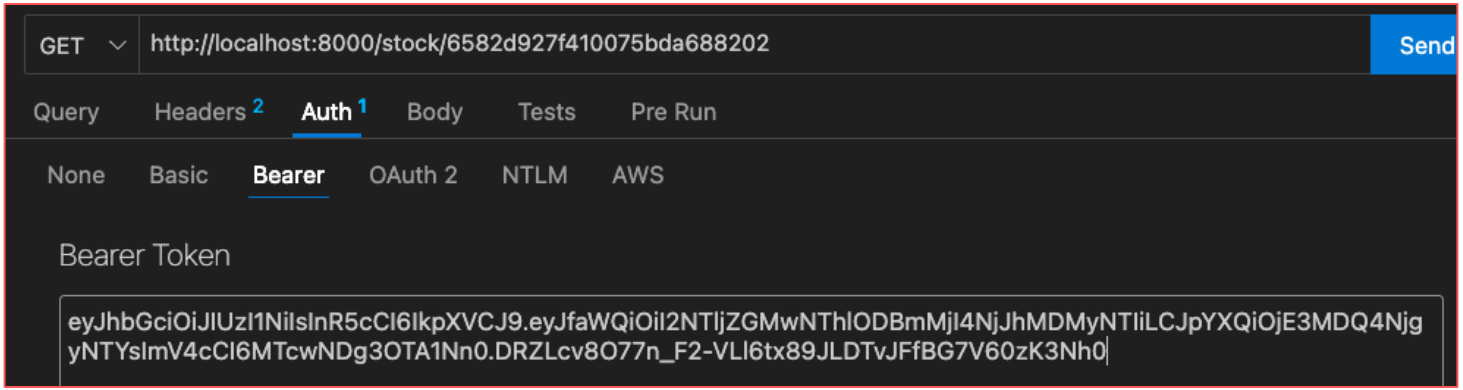
Achtung: Payload ist nicht verschlüsselt!

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☑ secret base64 encoded
```

Verwendung des Tokens bei Client Anfragen

Client’s GET Request an Webservice. JWT in HTTP Header:



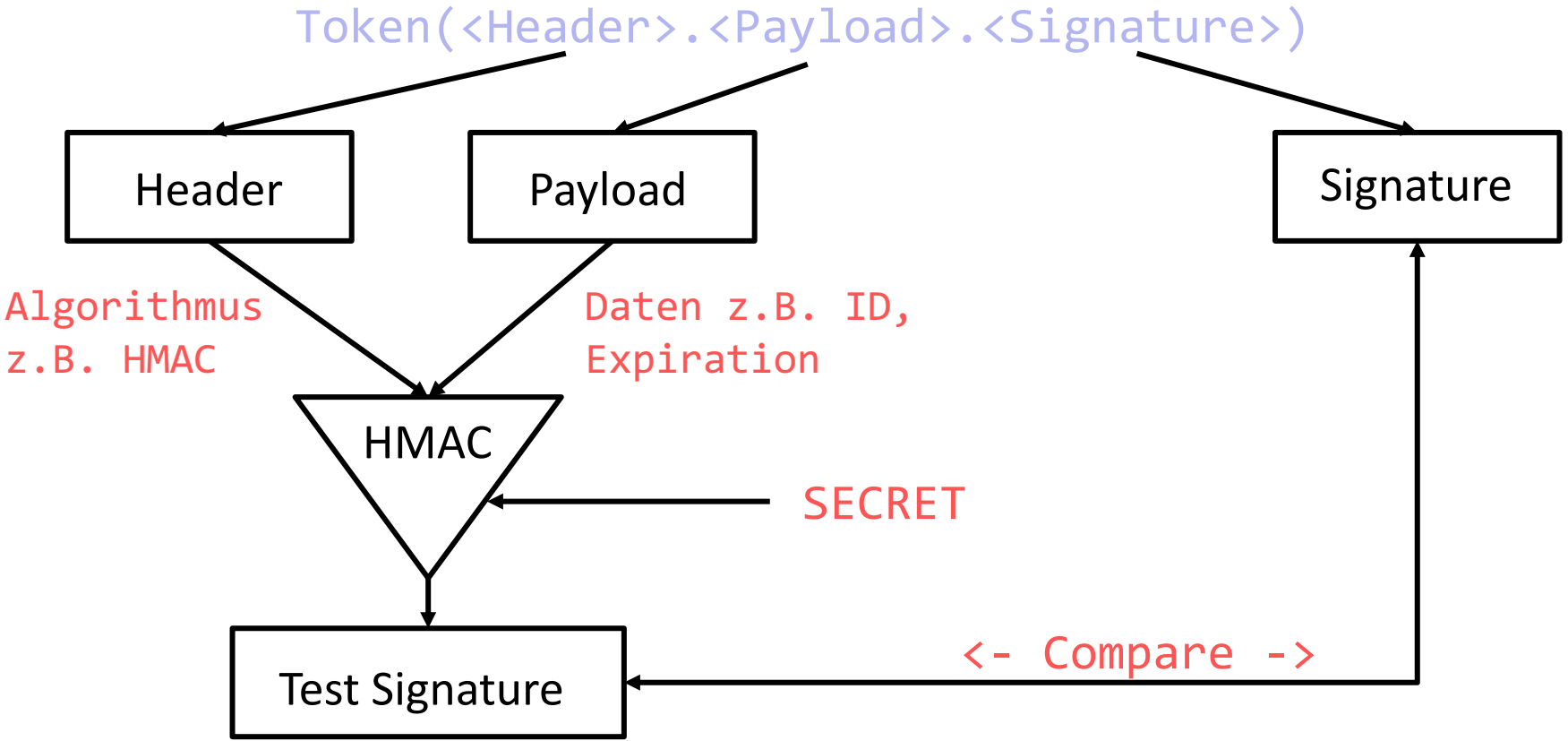

```
GET https://task-app-znkg.onrender.com/tasks
GET /tasks HTTP/1.1
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2MzI5OTE3MzU3MmI0NjFjNzRhOTIzNDIiLCJpYXQiOjE2NzEzNjM2MDR9.wL-IY_ox_UeW4wdu_k__8r
cSTi_Wriqy_QsZAmMdYRA
User-Agent: PostmanRuntime/7.29.2
Accept: */*
Postman-Token: 89258ade-d629-4421-9fd3-a726d9a38d28
Host: task-app-znkg.onrender.com
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

Schritt 3: Verification

Client sendet das Token bei Anfragen im Header mit. Das Token wird auf dem Server verifiziert und bei Erfolg eine Antwort zurückgegeben. Sonst Fehler.


```
{
  "category": "stock",
  "shareholders": [],
  "_id": "6582d927f410075bda688202",
  "name": "BASF",
  "price": 34.8,
  "high": 37.2,
  "low": 31.5
}
```

Token verifizieren



JSON Web Tokens (JWT)

Implementierung

Mongoose User Model

```
16  userSchema.pre("save", async function(next) {
17      if (!this.isModified("password")) return next();
18      // Hash password
19      this.password = await bcrypt.hash(this.password, 10);
20      next();
21  });
```

Pre Middleware Function

→ Beim Speichern das Passwort verschlüsseln (falls geändert)

```
npm i bcryptjs

npm i --save-dev @types/bcryptjs
```

https://mongoosejs.com/docs/middleware.html

jsonwebtoken

Weekly Downloads

2,020,182

Version	License
9.0.2	MIT
Unpacked Size	**Total Files**
43.5 kB	15
Issues	**Pull Requests**
115	41

Last publish

4 months ago

```
jwt.sign(payload, secretOrPrivateKey, [options, callback])
```

```
jwt.verify(token, secretOrPublicKey, [options, callback])
```

https://github.com/auth0/node-jsonwebtoken#readme

```
npm i jsonwebtoken
npm i --save-dev @types/jsonwebtoken
```

Anmelden

Benutzerdaten laden

```
29    const user: HydratedDocument<IUser> | null = await User.findOne({ email: email });
```

Passwort prüfen

```
35    const validPassword = await bcrypt.compare(password, user.password);
```

Token erstellen und signieren

```
41    const token: string = jwt.sign(
42        { _id: user._id },
43        process.env.TOKEN_SECRET || "tokentest",
44        { expiresIn: process.env.JWT_EXPIRES_IN || "1h" }
45    );
```

Endpoints absichern: Funktion verifyToken

Token aus Header lesen

```
54      if(req.headers.authorization && req.headers.authorization.startsWith("Bearer")) {
55          token = req.headers.authorization.split(" ")[1];
56      }
```

Token prüfen

```
63      try {
64          const verified = jwt.verify(token, process.env.TOKEN_SECRET || "tokentest");
65          console.log(verified);
66          next();
67      } catch (error) {
68          res.status(400).json({ message: "Invalid token!" });
69      }
```

Route absichern

```
11  router.get("/stock/:id", verifyToken, getStock)
```