

Erstelle mir eine CRUD API Anwendung mit Swagger und Express JS

Hier ist eine vollständige CRUD API Anwendung mit Express.js und Swagger. Diese Anwendung ermöglicht das Erstellen, Lesen, Aktualisieren und Löschen von Benutzern und bietet eine automatisch generierte API-Dokumentation mit Swagger UI.

## ✓ Projektstruktur

bash

Kopieren Bearbeiten

```
express-swagger-crud/
├─ index.js           # Einstiegspunkt
├─ routes/
│   └─ users.js       # Benutzer-CRUD-Routen
├─ swagger/
│   └─ swagger.yaml   # Swagger-Definition
├─ swagger-ui.js      # Swagger Setup
└─ package.json
```

## 📦 1. Projekt einrichten

bash

Kopieren Bearbeiten

```
mkdir express-swagger-crud
cd express-swagger-crud
npm init -y
npm install express swagger-ui-express yamljs
```

## 2. index.js

js

 Kopieren  Bearbeiten

```
const express = require('express');
const userRoutes = require('./routes/users');
const swaggerUi = require('swagger-ui-express');
const swaggerDocument = require('./swagger-ui');

const app = express();
const PORT = 3000;

app.use(express.json());
app.use('/api/users', userRoutes);
app.use('/api-docs', swaggerUi.serve, swaggerUi.setup(swaggerDocument));

app.listen(PORT, () => {
  console.log(`🚀 Server läuft auf http://localhost:${PORT}`);
  console.log(`📘 Swagger UI verfügbar unter http://localhost:${PORT}/api-docs`);
});
```



### 3. routes/users.js

js

Kopieren Bearbeiten

```
const express = require('express');
const router = express.Router();

let users = [
  { id: 1, name: 'Max Mustermann' },
  { id: 2, name: 'Erika Musterfrau' }
];
let nextId = 3;

// Alle Benutzer abrufen
router.get('/', (req, res) => {
  res.json(users);
});

// Benutzer nach ID abrufen
router.get('/:id', (req, res) => {
  const user = users.find(u => u.id === parseInt(req.params.id));
  if (!user) return res.status(404).json({ message: 'Benutzer nicht gefunden' });
  res.json(user);
});

// Benutzer erstellen
router.post('/', (req, res) => {
  const user = { id: nextId++, name: req.body.name };
  users.push(user);
  res.status(201).json(user);
});

// Benutzer aktualisieren
router.put('/:id', (req, res) => {
  const user = users.find(u => u.id === parseInt(req.params.id));
  if (!user) return res.status(404).json({ message: 'Benutzer nicht gefunden' });
  user.name = req.body.name;
  res.json(user);
});

// Benutzer löschen
router.delete('/:id', (req, res) => {
  const index = users.findIndex(u => u.id === parseInt(req.params.id));
  if (index === -1) return res.status(404).json({ message: 'Benutzer nicht gefunden' });
  users.splice(index, 1);
  res.status(204).send();
});

module.exports = router;
```





#### 4. swagger-ui.js

js

Kopieren Bearbeiten

```
const YAML = require('yamljs');  
const swaggerDocument = YAML.load('./swagger/swagger.yaml');  
module.exports = swaggerDocument;
```



#### 5. swagger/swagger.yaml

yaml

Kopieren Bearbeiten

```
openapi: 3.0.0  
info:  
  title: Benutzer API  
  version: 1.0.0  
  description: Eine einfache CRUD-API mit Express und Swagger  
  
servers:  
  - url: http://localhost:3000  
  
tags:  
  - name: Users  
    description: Benutzerverwaltung
```

1/3

```
paths:
  /api/users:
    get:
      tags: [Users]
      summary: Alle Benutzer abrufen
      responses:
        '200':
          description: Erfolgreich
          content:
            application/json:
              schema:
                type: array
                items:
                  $ref: '#/components/schemas/User'

    post:
      tags: [Users]
      summary: Neuen Benutzer erstellen
      requestBody:
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/UserInput'
      responses:
        '201':
          description: Benutzer erstellt
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/User'
```

2/3

```
/api/users/{id}:
  get:
    tags: [Users]
    summary: Benutzer nach ID abrufen
    parameters:
      - name: id
        in: path
        required: true
        schema:
          type: integer
    responses:
      '200':
        description: Benutzer gefunden
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/User'
      '404':
        description: Nicht gefunden

  put:
    tags: [Users]
    summary: Benutzer aktualisieren
    parameters:
      - name: id
        in: path
        required: true
        schema:
          type: integer
    requestBody:
      required: true
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/UserInput'
    responses:
      '200':
        description: Benutzer aktualisiert
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/User'
```

3/3

```
delete:
  tags: [Users]
  summary: Benutzer löschen
  parameters:
    - name: id
      in: path
      required: true
      schema:
        type: integer
  responses:
    '204':
      description: Benutzer gelöscht
```

components:

schemas:

User:

type: object

properties:

id:

type: integer

example: 1

name:

type: string

example: Max Mustermann

UserInput:

type: object

required:

- name

properties:

name:



type: string

example: Max Mustermann

+4

## Anwendung starten

```
bash
```

 Kopieren  Bearbeiten

```
node index.js
```

- **API Testen:** <http://localhost:3000/api/users>
- **Swagger UI:** <http://localhost:3000/api-docs>

## Optional

Möchtest du zusätzlich:

- MongoDB oder SQLite anbinden?
- TypeScript nutzen?
- JWT Authentifizierung hinzufügen?

Ich kann dir das auch als GitHub-Projektvorlage oder Zip-Datei bereitstellen – sag einfach Bescheid.