#node
#postgres
#passportjs
#jwt
#vue
#vuex

Christopher Praas, Martin List, Kristof Kind

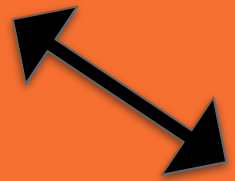
03.12.2018

???

API

Frontend

Express



PostgreSQL

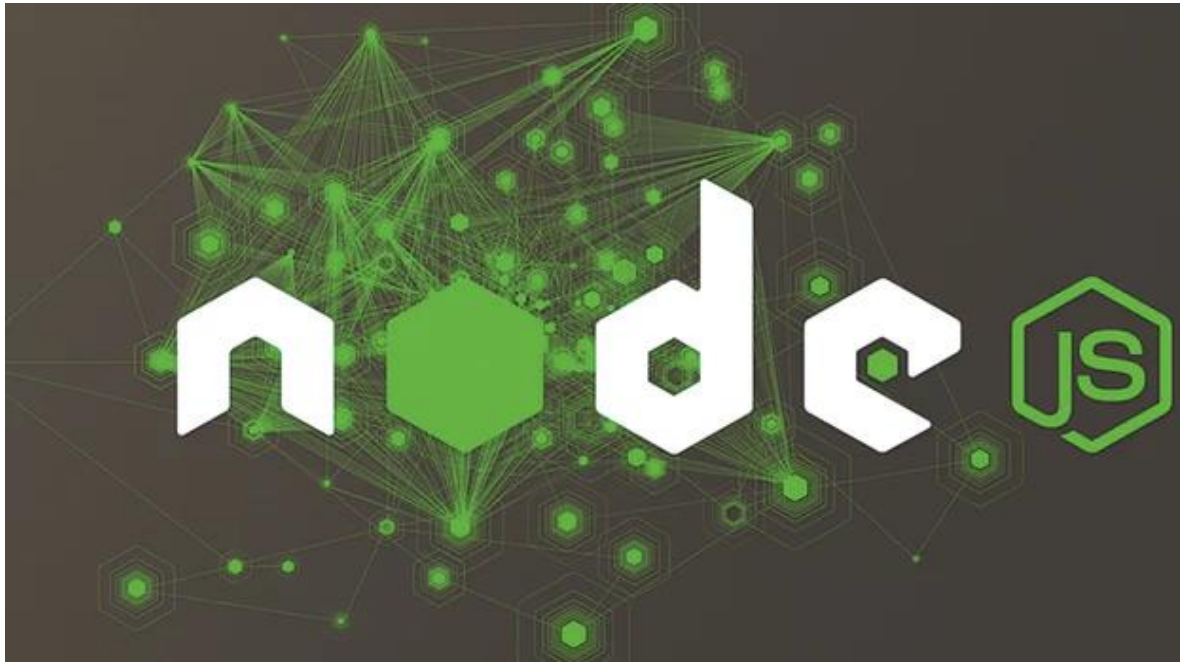
Wie gehen wir vor?

Vorgehen

1. Node.js

Node.js

- Download node.js [https://nodejs.org/en/download/]



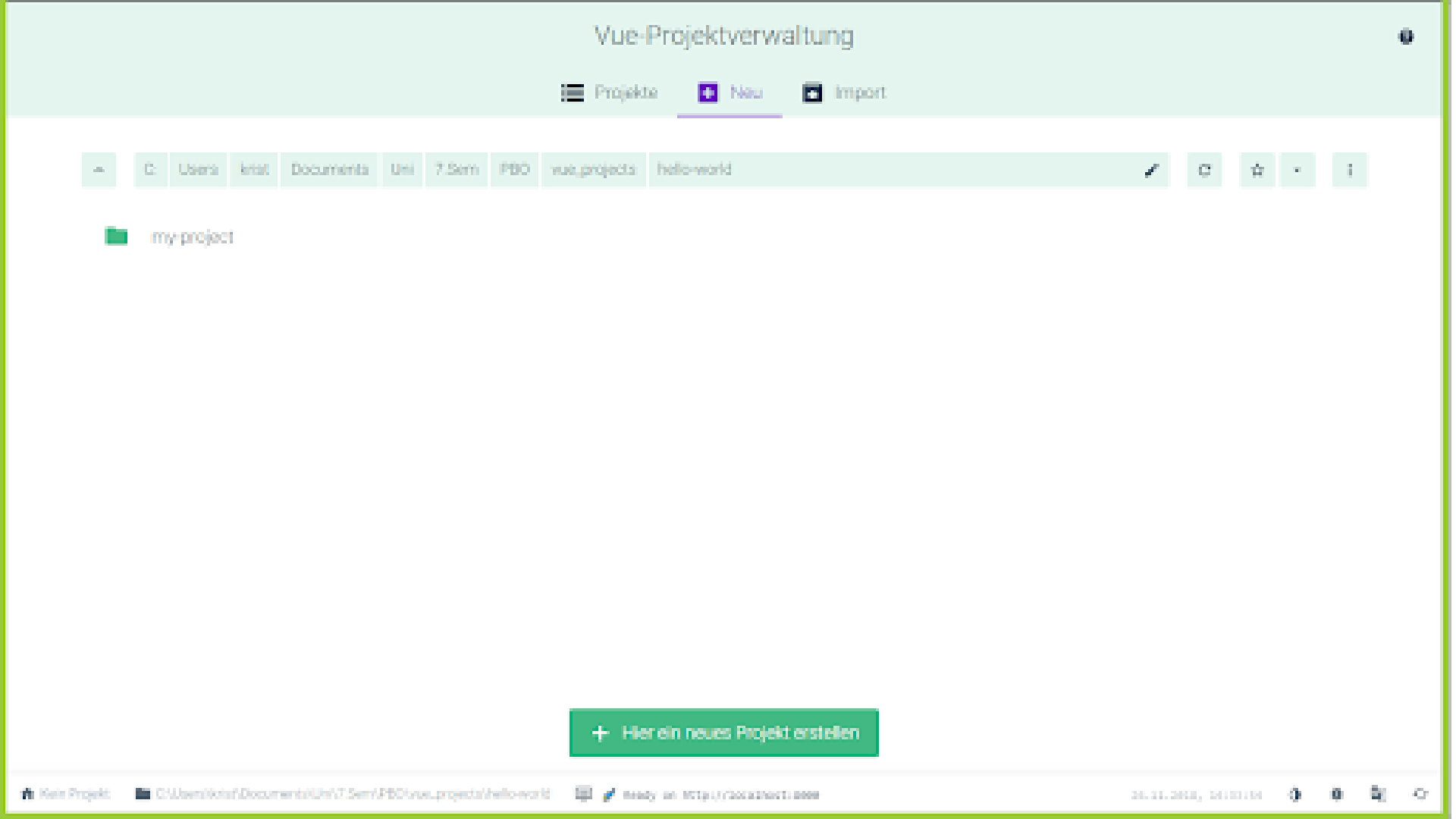
Vorgehen

1. Node.js
2. Vue CLI

Projekterstellung mit Vue CLI

- `npm install -g vue-cli`
  - `vue ui`
- Express generator
  - `npm install express-generator -g`
  - `express testapp`

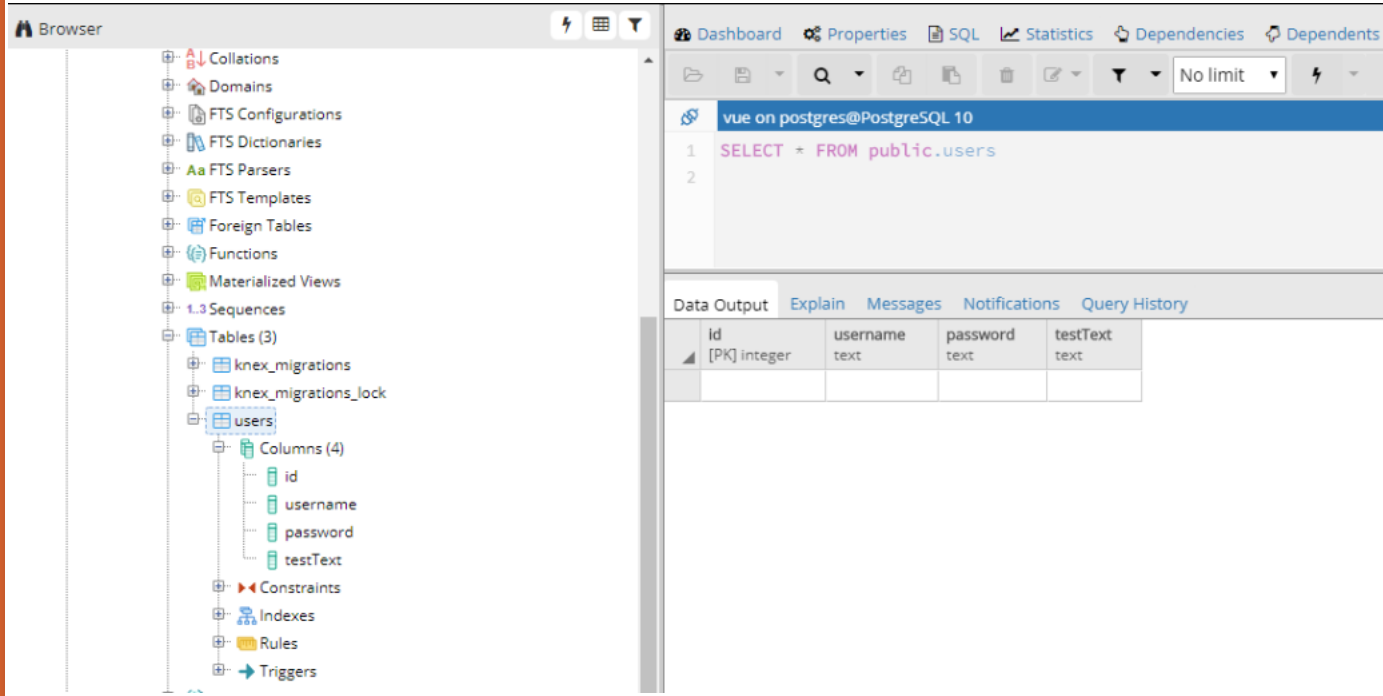
Express


Vue-Projektverwaltung
Projekte
Neu
Import
C:
Users
krist
Documents
Uni
7.Sem
PBO
vue_projects
hello-world
my project
Hier ein neues Projekt erstellen
Kein Projekt:
C:\Users\krist\Documents\Uni\7.Sem\PBO\vue_projects\hello-world
Ready on: http://localhost:8080

Vorgehen

1. Vue CLI
2. Node.js
3. PostgreSQL

PostgreSQL



Vorgehen

1. Vue CLI
2. Node.js
3. PostgreSQL
4. Dependencies

Dependencies (1)

- Entweder über `vue ui`
- Oder terminal über `npm install [dependency]`

- Wir verwenden (auswahl)
  - Express → Framework für WebAPI
  - Knex → Query builder für SQL
  - Passport (-jwt)→ Authentication
  - Jsonwebtoken → Authentication
  - Axios → http client mit promises

Vorgehen

1. Vue CLI
2. Node.js
3. PostgreSQL
4. Dependencies

Dependencies (2)

```
npm install axios bluebird body-parser
bootstrap-vue connect-history-api-fallback
dotenv express http-errors jquery
jsonwebtoken knex morgan passport passport-
jwt pg secure-password vue-axios
```

Vorgehen

1. Vue CLI
2. Node.js
3. PostgreSQL
4. Dependencies

Passport

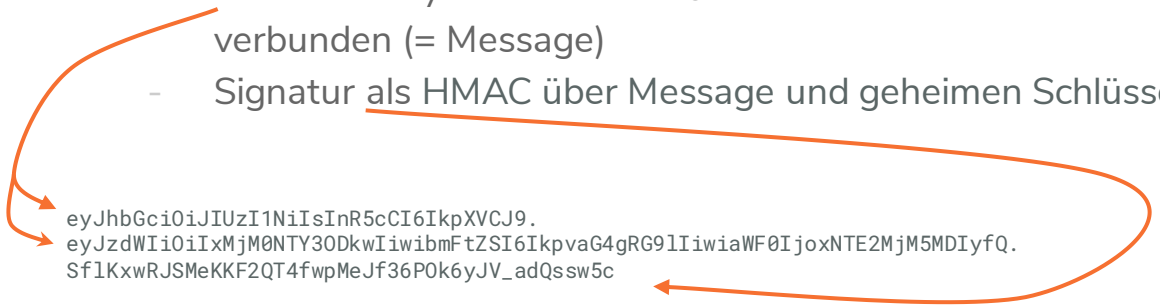
- Authentifizierung für Express-basierte Web-Applikationen
- Kernmodul stellt Infrastruktur zur Verfügung
- Implementierung über *Strategies*
  - 500+ Authentifizierungs-Methoden
  - username + password, Facebook, Twitter, JWT, OAuth
  - besitzen eigenes npm package
- `$ npm install passport`
- `$ npm install passport-<strategy>`
- ⚠ Passwortspeicherung, Hashing werden nicht von Passport übernommen

Vorgehen

1. Vue CLI
2. Node.js
3. PostgreSQL
4. Dependencies

JSON Web Token

- Access-Token welches verifizierbare Claims zwischen Client und Server ermöglicht (RFC 7519)
- Besteht aus Payload, Header und Signature
- ⚠ Codierte JWT lassen sich auslesen
- Header + Payload sind Base64 kodiert und mit "." verbunden (= Message)
- Signatur als HMAC über Message und geheimen Schlüssel



Vorgehen

1. Vue CLI
2. Node.js
3. PostgreSQL
4. Dependencies
5. API

API

- API erstellen mittels express

```
8   /* easy get example */
9   router.get("/", function(req, res) {
10    res.send("Hello World");
11  });
12
13  /* post example */
14  router.post("/", function(req, res) {
15    if (typeof req.body.username !== "undefined") {
16      res.send("Hello " + req.body.username);
17    } else {
18      res.status(400).send("There is no username!");
19    }
20  });
21
```

Vorgehen

1. Vue CLI
2. Node.js
3. PostgreSQL
4. Dependencies
5. API

API

- `Knex init`
- `knex migrate:make migration_name`
- `knex migrate:latest --env production`
- `knex migrate:rollback`

Vorgehen

1. Vue CLI
2. Node.js
3. PostgreSQL
4. Dependencies
5. API

API

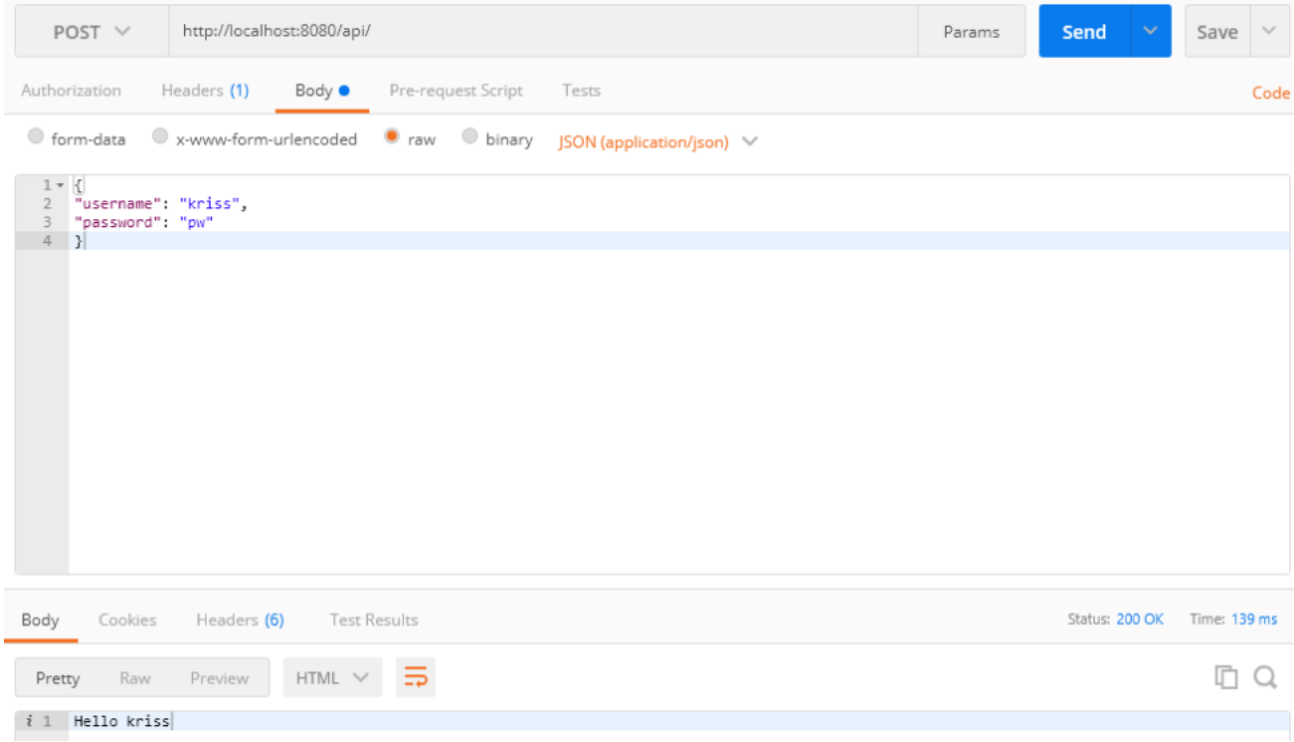
```
{
  "name": "pbovue",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "start": "node ./bin/www",
    "serve": "vue-cli-service serve",
    "build": "vue-cli-service build",
    "lint": "vue-cli-service lint"
  },
  "dependencies": {
```

Vorgehen

1. Vue CLI
2. Node.js
3. PostgreSQL
4. Dependencies
5. API

API

- Testen mittels Restlet oder Postman



Vorgehen

1. Vue CLI
2. Node.js
3. PostgreSQL
4. Dependencies
5. API
6. Frontend

Frontend

Vorgehen

1. Vue CLI
2. Node.js
3. PostgreSQL
4. Dependencies
5. API
6. Frontend

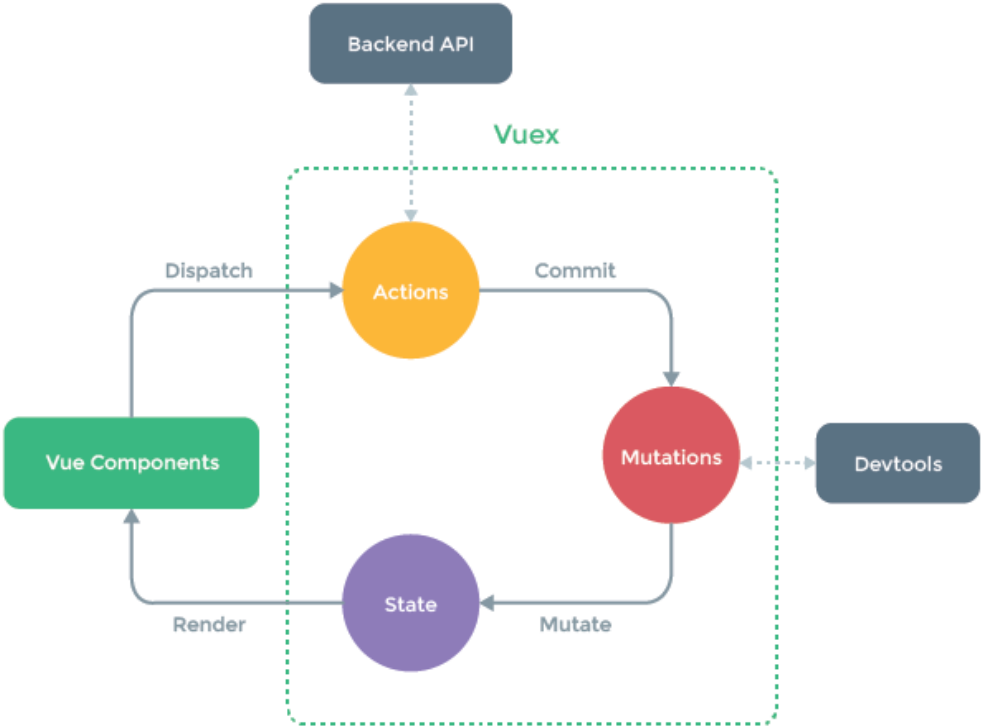
Frontend

```
<template>
  <b-container id="loginForm">
    <div>
      <b-form @submit="onSubmit">
        <b-form-input
          id="inputUsername"
          v-model="username"
          type="text"
          placeholder="Username"
          :state="usernameState"
        />
        <b-form-input
          id="inputPassword"
          v-model="password"
          type="password"
          placeholder="Password"
          :state="passwordState"
        />
        <b-button
          id="submitButton"
          type="submit"
        >Login</b-button>
      </b-form>
    </div>
  </b-container>
</template>
```

Vorgehen

1. Vue CLI
2. Node.js
3. PostgreSQL
4. Dependencies
5. API
6. Frontend
7. Vuex

Frontend - Vuex



Vorgehen

1. Vue CLI
2. Node.js
3. PostgreSQL
4. Dependencies
5. API
6. Frontend
7. Vuex
8. Quellcode

Quellcode

https://github.com/Schnitzel128/PBO-Vue

Vuehu!