



TWANIMAL

SOZIALES NETZWERK FÜR TIERE

Molham Al Khodari, Sarah Schelle, Salma Alkhaiyal, Anna-Lisa Merkel, Luca Voges

GLIEDERUNG

1. Teamvorstellung
2. Projektidee
3. Verwendete Schnittstellen / API
4. Zusammenarbeit in der Gruppe
5. CSS-Präprozessor
6. Typescript oder Javascript
7. Datenbank und ORM
8. Ordnerstruktur
9. Mockups
10. Live-Demo
11. Anmerkungen zum Kurs
12. Abschluss und Tools





1. TEAMVORSTELLUNG

Molham Al Khodari

Alter: 25

Hobbies: Sport, Videogames

Skills: C++, java, php, javascript,
HTML, CSS, SASS

Sarah Schelle

Alter: 20

Hobbies: Singen, Fotografie,
Videospiele

Skills: JavaScript,
HTML, CSS, SASS, php, C++

Salma Alkhaiyal

Alter: 25

Hobbies: Sport, Spazieren

Skills: HTML, CSS, Javascript,
SASS, php, C++

Luca Voges

Alter: 21

Hobbies: Wandern, Spiele & Essen

Skills: Vue.JS, JS, HTML, CSS &
SASS, TS, Java, C++, Kotlin, Python

Anna-Lisa Merkel

Alter: 20

Hobbies: Spiele, kreatives Zeug

Skills: JavaScript, HTML, CSS,
SASS, php, C++

2. PROJEKTIDEE

Soziales Netzwerk für Haustiere auf Basis von Twitter

Kernfunktionen

- Erstellung, Abrufen und Interaktion von Profilen und Beiträgen
- Kontakte aufbauen durch Empfehlungen
- Kommentieren von Beiträgen
- Finden von Beiträgen und Profilen
- Schnittstelle für mögliche externe Anwendungen
- Reaktionen teilen durch Gifs und Sticker

3. VERWENDETE SCHNITTSTELLEN / API

- Eigene Twanimal-Schnittstelle
 - Nutzer: Anmelden, Registrieren, Abrufen, (Ent-)folgen, Beiträge
 - Beiträge: Erstellen, Löschen, Abrufen, (Ent-)liken, Antworten
 - Timeline mit Beiträgen von Profilen denen man folgt
- Externe Schnittstelle von Giphy
 - Abrufen von beliebten Gifs und Stickern
 - Suche nach bestimmten Gifs und Stickern
 - Zufällige Gifs entsprechend Auswahl

```
router.post(
  "/user/:id/unfollow",
  userService.authenticateMiddleware,
  userService.getUserMiddleware,
  userService.unfollowUserMiddleware,
  async (req, res) => res.json(await userService.exportUser(req.data, false, req.user))
)

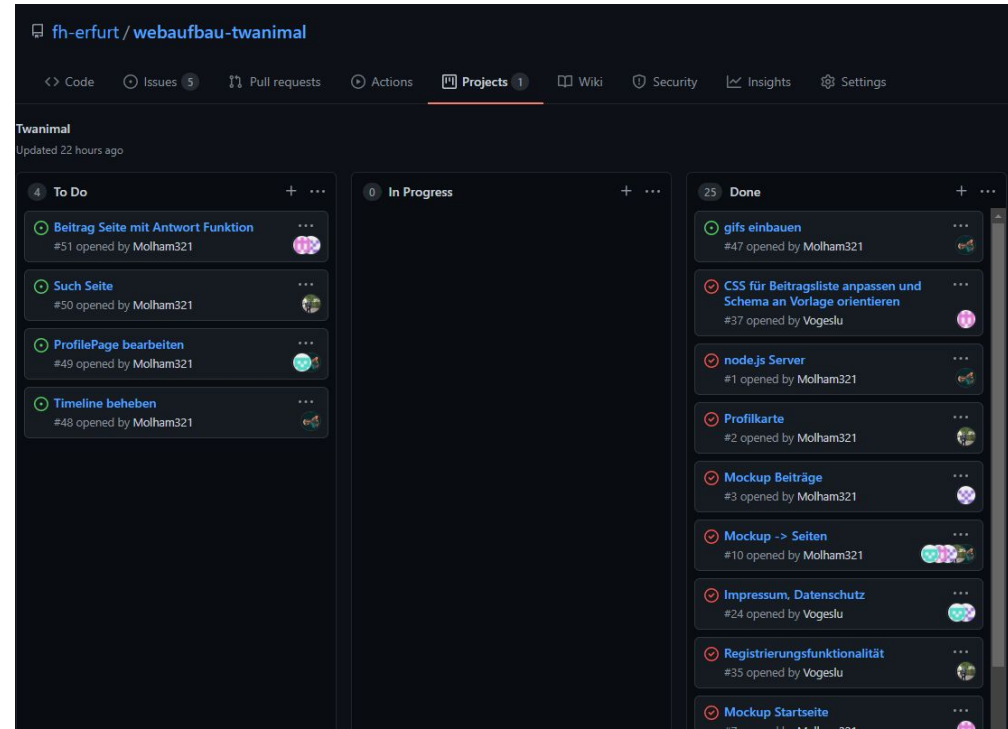
router.all(
  "/user/:id/posts",
  userService.getAuthenticatedUserMiddleware,
  userService.getUserMiddleware,
  paginationResultService.parsePaginationResultMiddleware(20),
  postService.getPostsFromUserMiddleware,
  (req, res) => res.json(req.paginationResult)
)

router.all(
  "/home-timeline",
  userService.authenticateMiddleware,
  paginationResultService.parsePaginationResultMiddleware(20),
  postService.getHomeTimelineMiddleware,
  (req, res) => res.json(req.paginationResult)
)

router.all(
  "/suggestions",
  userService.authenticateMiddleware,
  paginationResultService.parsePaginationResultMiddleware(5, 20),
  userService.getUserSuggestionsMiddleware,
  (req, res) => res.json(req.paginationResult)
)
```

4. ZUSAMMENARBEIT IN DER GRUPPE

- Wöchentliche Meetings mit
 - Vorstellung der erledigten Aufgaben
 - Verteilung von neuen Aufgaben
 - Besprechung von Ideen und Vorschlägen
 - “Team Programming”
- Weitere Meetings in kleineren Gruppen
 - Pair Programming





5. CSS-PRÄPROZESSOR

- Komplette Integration von SCSS im gesamten Projekt
 - Wechsel von CSS auf SCSS im frühen Projektablauf
 - Besser strukturierter Aufbau dank u.a. Nesting
 - Verwenden von integrierten Funktionen, wie darken, lighten
 - Einbinden von SCSS-Modulen für u.a. Buttons
-
- Einbinden von SCSS Klassen in React-Komponenten als Module

```
.box {  
  background: #fff;  
  margin: 0 auto;  
  max-width: 900px;  
  border-radius: 4px;  
  box-shadow: 0 1px 3px rgb(0 0 0 / 6%), 0 1px 2px rgb(0 0 0 / 12%);  
  padding: 25px;  
  position: relative;  
  
  .close {  
    position: absolute;  
    top: 5px;  
    right: 5px;  
    width: 40px;  
    height: 40px;  
    border-radius: 20px;  
    display: flex;  
    align-items: center;  
    justify-content: center;  
    color: #3c3c3c;  
    transition: background .2s ease;  
    cursor: pointer;  
  
    &:hover {  
      background: #f1f1f1;  
    }  
  }  
  
  h3 {  
    color: #252525;  
    font-size: 18px;  
    font-weight: 600;  
    margin: 0 0 20px 0;  
  }  
}
```

6. TYPESCRIPT ODER JAVASCRIPT

- Verwendung von TS im Backend-System von Twanimal
 - Verständliche Type Annotations
 - Schnelles Erkennen von Fehlern
 - Verbesserte Lesbarkeit
 - Globales Verwenden von import, export, async, await ohne Probleme
- Verwendung von normalem Javascript in JSX-Klassen im
- Frontend von Twanimal
 - Leichter für Einsteiger
 - Kein Stress durch fehlende Type Annotations

```
interface PostExport {
  id: number
  createdBy: UserExport
  createdAt: number
  text: string
  attachments: any
  likeCount: number
  hasLiked?: boolean
  replyTo?: PostExport | number
  repostOf?: PostExport | number
}

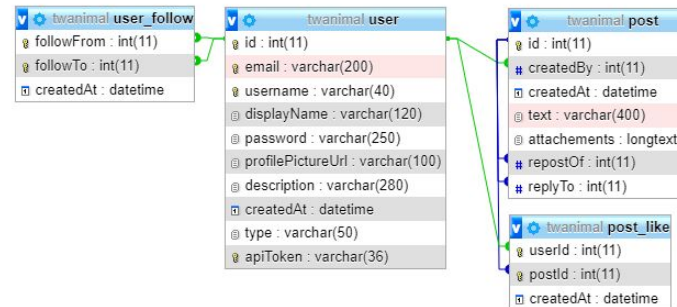
async function exportPost(
  post: Post,
  user: User = null,
  isNested: boolean = false
): Promise<PostExport> {
  const createdBy = await prisma.user.findUnique({
    where: {
      id: post.createdBy,
    },
  })
}
```


7. DATENBANK UND ORM

- Datenbank ist klein... aber wirksam
- Integration im Twanimal-Backend mit dem ORM

Prisma, spezialisiert auf Typescript

- Sichere Type Annotations
- Einfache Datenbankmigration ohne Erstellungsskripte
- Einfache Anwendung dank Features von ECMAScript 2017 (await, async)



```

model User {
  id          Int           @id @default(autoincrement())
  email       String        @unique @db.VarChar(200)
  username    String        @unique @db.VarChar(40)
  displayName  String        @db.VarChar(120)
  password    String        @db.VarChar(250)
  profilePictureUrl String?  @db.VarChar(100)
  description  String?       @default("") @db.VarChar(280)
  createdAt   DateTime       @default(now()) @db.DateTime(0)
  type        String         @db.VarChar(50)
  apiToken    String         @unique @db.VarChar(36)
  post        Post[]         @relation("postToUser")
  postLike    PostLike[]     @relation("postLikeToUser")
  followFrom  UserFollow[]   @relation("userFollowFrom")
  followTo    UserFollow[]   @relation("userFollowTo")

  @@map("user")
}
  
```

8. ORDNERSTRUKTUR


Frontend (React)

```
> public
  > src
    > assets
      > css
      > fonts
      > images
    > components
    > routes
    > services
  JS App.js
  JS App.test.js
  JS config.js
  # index.css
  JS index.js
  logo.svg
  JS reportWebVitals.js
  JS setupTests.js
  .gitignore
  .prettierrc
  package.json
  README.md
  yarn.lock
```

Backend (Node.JS + TS)

```
> dist
> node_modules
  > prisma
    > migrations
    schema.prisma
  > public\images
  > src
    > routes
    > services
  TS config.ts
  TS custom.d.ts
  TS index.ts
  > upload
  .env
  .gitignore
  package-lock.json
  package.json
  README.md
  tsconfig.json
  yarn.lock
```

9. MOCKUPS



Log in to Twitter

Phone, email, or username

Password

Log in

Forgot password? Sign up for Twitter

Registrierung

Erstelle einen neuen Account bei Twanimal

Name des Haustieres

Nutzurname

Email

Password Password wiederholen

Registrieren

Doch schon Teil der Crew? Hier geht's zur Anmeldung!

Log in

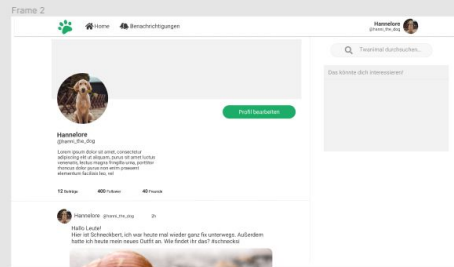
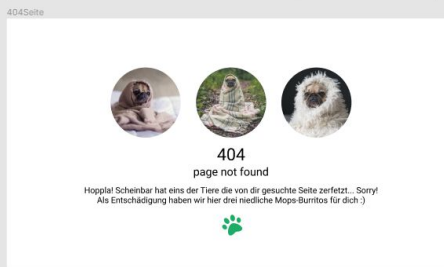
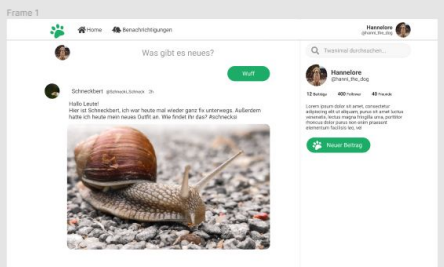
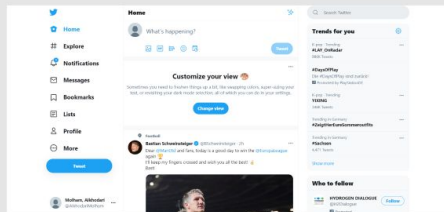
Melde dich bei Twanimal an

Email

Password

Log in

Noch nicht Teil der Crew? Hier geht's zur Registrierung



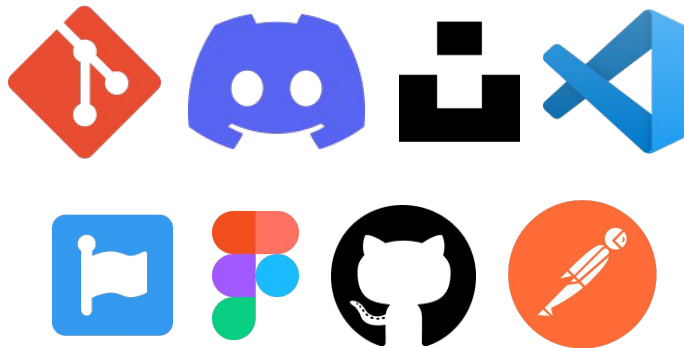
10. LIVE-DEMO



11. ANMERKUNGEN ZUM KURS

- Wöchentliche Aufgaben, wie Einbinden einer externen API und weiteres weitaus früher ankündigen und nicht im Wochentakt (erzeugt unnötigen Stress)
- Checkliste für Meilensteine für Abgabe ähnlich wie in dynamische Webprogrammierung (3. Semester)

12. ABSCHLUSS UND TOOLS



Vielen Dank für Ihre Aufmerksamkeit ♥

Für weitere Fragen stehen wir Ihnen gerne zur Verfügung

Link zur Live-Seite: twanimal-live.vogeslu.de