Web Engineering

Dr. Michael Lesniak
mlesniak@micromata.de

13. Mai 2019

MICROMATA

Heute...

- Organisatorisches
- Ausblick
- Review Übungsaufgaben
- REST & GraphQL (mit Demo)
- Code
  - Backend: Posts und Kommentare
  - Frontend: Interaktivität, Eigene Post-Seite & Routing

---------- 5 Minuten Pause ----------

- Übung

Organisatorisches...

- Projektaufgabe – ihr dürft kreativ sein ;-) -> #project-ideas
- (Entscheidung am Ende immer noch bei mir)

- Beteiligung bei Feedback zu Gruppenaufteilung gering ¯_(ツ)_/¯

Ausblick (Was kommt noch?)

- Authentifizierung im Frontend und Backend
- Saubere Paketierung des Frontends (und damit z. B. Modulsystem ES6)
- Deployment in eine Cloud-Umgebung
- Refactorings

- Optional
  - Logging und Monitoring
  - Git Workflows
  - Suche

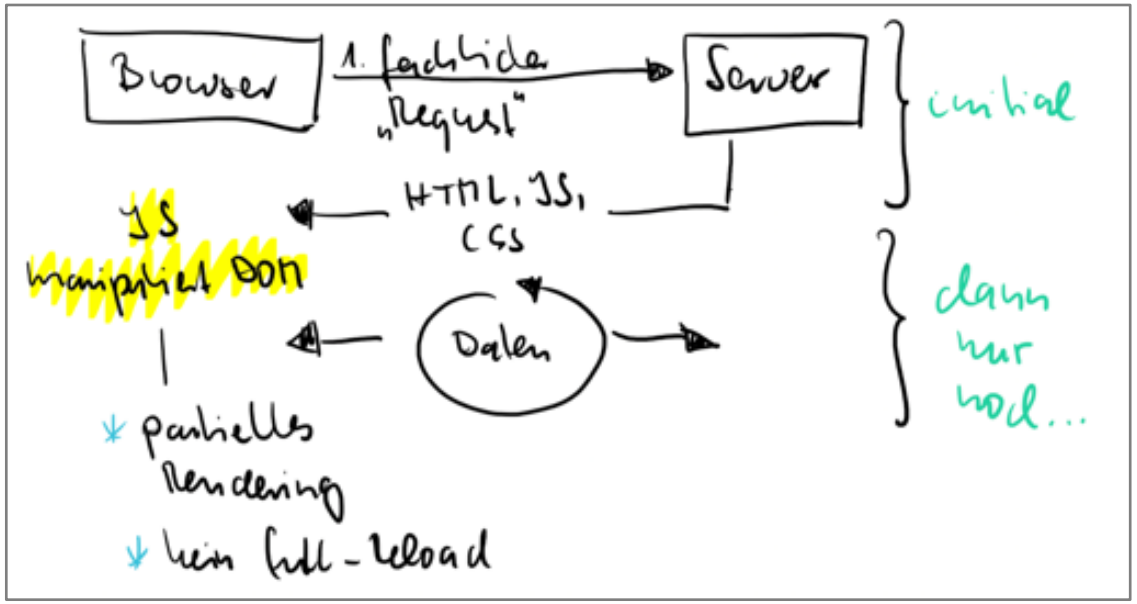
Review Übungsaufgaben

- Backend
  - Zusatzmethode: nur die letzten 5 aktuellsten Einträge
  - POJO braucht einen Zeitstempel
  - Zur Vereinfachung: `@Query` Annotation und `nativeQuery = true`

- Frontend
  - Nummerierung und Daten an jedem Post anzeigen
  - Eingabefeld aus letzter Übung einbauen und Post-Eingabe ermöglichen (ohne React!)
  - Click-Handler für Click auf einzelne Zeile mit „Geklickt Ausgabe“ in der Browser-Konsole

REST und GraphQL

- Motivation: Backend liefert nur Daten
- Daten müssen abgefragt werden
- Zu schreibende Daten müssen definiert werden
- Verschiedene technische Ansätze denkbar (-> Tradeoffs)



Siehe Vorlesung vom 2019-05-06

REST

- Ressourcen werden durch URI referenziert
- Endpunkte beschreiben die zu nutzende Resource (/api/post, /api/post/)

- Hoher Freiheitsgrad:
  - Verschiedene Konventionen bei Namen und Endpunkten, aber keinen Standard
  - Wichtig: Konsistenz

- HTTP Verben definieren CRUD-Operation
  - **C**REATE mit POST
  - **R**EAD mit GET
  - **U**PDATE mit PUT
  - **D**ELETE mit DELETE

GraphQL

- Nachteile von REST u.a.
  - ggf. viele Requests
  - Ausgabe-Format nicht wohl-definiert

- GraphQL
  - Definiert Datentypen und Struktur über ein Schema
  - erlaubt viele Abfragen zu einer komplexen Abfrage zusammenzufassen (dynamische Attributauswahl)
  - Ausgabe-Format entspricht Abfrage-Format
  - Erlaubt Updates in Form von Lesen (*query*) und Schreiben (*mutation*)

GraphQL Explorer Demo https://developer.github.com/v4/explorer/

Tradeoffs

- Was würde ich heute verwenden?
- Kommt darauf an… Case-by-Case-Entscheidung
- Schneller und komfortabler vs. weiter-verbreitet
- Leitfragen z. B.
  - Brauche ich (absehbar) die Flexibilität von GraphQL
  - Wie viele (verschiedene) Konsumenten meines Backends gibt es?
  - Wie flexibel sind sie?

- Strategische Gründe: Warum REST in der Vorlesung?

Code-Walkthrough

- Neben Posts jetzt auch Kommentare im Backend persistieren
- Details eines Post (Titel und geschachtelte Kommentare) anzeigen
  - Refactoring von Funktionalität in Komponenten
  - Routing

- Heute in der Vorlesung nicht:
- Eingabe-Komponente

Übung

- Commit für Commit auschecken und verstehen (ggf. fragen!)
- Backend
  - Separates Comment-Objekt statt Kommentare in Posts speichern
  - Post-Objekt soll dedizierte URL haben
  - Refactoring – überlegen, was sinnvoll sein könnte (YMMV)
- Frontend
  - Anschauen von https://jgthms.com/web-design-in-4-minutes/
  - CSS für Detailseite bauen
  - Eigene Komponente für Eingabe eines Textfeldes unter `/post/new` bauen (siehe https://reactjs.org/docs/forms.html)