Otto-Friedrich-Universität Bamberg
Lehrstuhl für Medieninformatik

Web-Technologien, Sommersemester 2025

Teilleistung 4

Abgabe der Lösungen

- *Abgabetermin.* **15. Juli 2025, 23:55 Uhr**
- *Abgabemodus.* Ordner als ZIP-Datei im Virtuellen Campus im entsprechenden VC-Kurs des Moduls. Bei einer Bearbeitung im Team, genügt die Abgabe durch ein einzelnes Teammitglied. Die zuletzt hochgeladene Abgabe wird bewertet.
- *Dateiformat der Abgabe.* Die Abgabe erfolgt durch den Upload des ZIP-Archivs, das alle notwendigen Dateien beinhalten soll.
- *Update der Lösung.* Bis zur oben angegebenen Deadline könnt Ihr Eure Lösung beliebig oft durch eine neue (korrigierte) Fassung ersetzen. Zuvor hochgeladene Lösung werden dabei überschrieben. Wir haben keine Möglichkeit, alte Fassungen wiederherzustellen!

Hinweise

- Die vollständige und korrekte Bearbeitung einer Aufgabe ergibt die volle Punktzahl dieser Aufgabe.
- Für diese Teilleistung erwarten wir einen Bearbeitungsaufwand von insgesamt etwa 45 Stunden, da wir von vier Personen pro Team ausgehen. Dies ist als Richtwert zu verstehen und keinesfalls als Unter- oder Obergrenze.
- In Verdachtsfällen behalten wir uns vor, Plagiate von der Bewertung auszunehmen. Wir werden in einem solchen Fall Kontakt mit den Betroffenen aufnehmen.
- Programmieraufgaben sollen in nachvollziehbar kommentierter Form im Quelltext abgegeben werden. Dabei dürfen keine anderen als die angegebenen Technologien verwendet werden. Nicht lauffähige Programme können nicht bewertet werden. Der Abgabe soll eine README-Datei hinzugefügt werden, die beschreibt, wie man die Lösung ausführen kann.
- Bei der Auswahl der Medien sind keine Urheberrechte zu berücksichtigen, da eine Veröffentlichung der Teilleistung nicht vorgesehen ist.

Aufgabe: Erstellen einer Website für Erfahrungsberichte der Besucher des Rock am See Festivals (30 Punkte)

Die Veranstalter des Rock am See Festivals möchten ihren Besucher:innen eine Möglichkeit geben, ihre Erfahrungen zu teilen und Feedback zu geben. Dafür soll eine neue Single-Page-Application erstellt werden, auf der Berichte gepostet und angezeigt werden können (vgl. Abbildung 1). Um Datenverlust zu verhindern, sollen die Berichte zunächst lokal im localStorage (`https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage`) gespeichert werden. (Eine Anbindung an ein Backend mit Datenbank ist erst in Zukunft geplant.)

Es wurde entschieden, dass der Webauftritt mithilfe von *Svelte* und *Bootstrap* erstellt werden soll. Beides ist bereits im Euch zur Verfügung gestellten Template eingebunden. Mit `npm run dev` könnt ihr den Dev-Server von *Vite* starten. Die im Template enthaltenen `.svelte`-Dateien sollen lediglich den Einstieg erleichtern und nicht mit abgegeben werden.

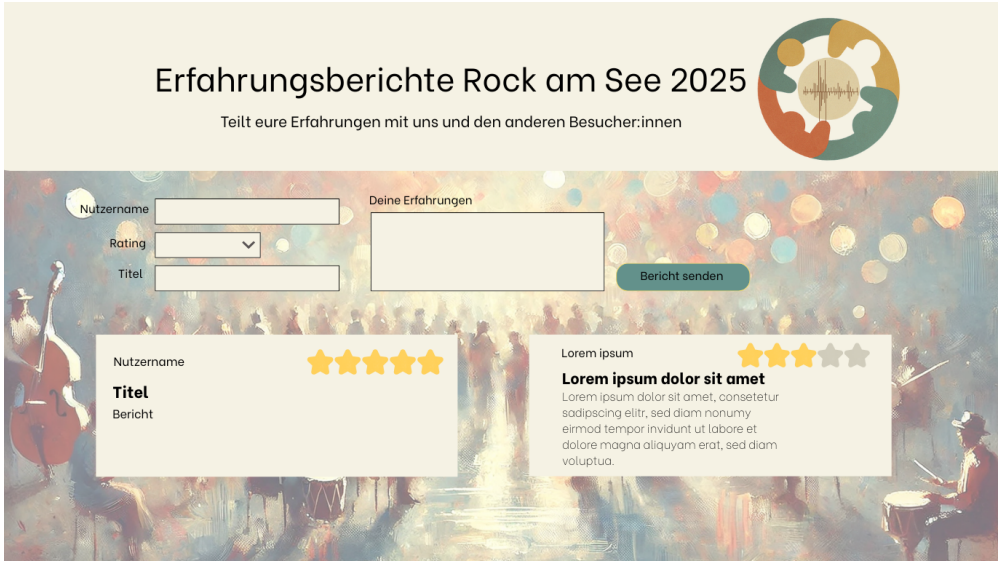

Abbildung 1: Mockup der Anwendung

Anforderungen an Struktur und Inhalt

- Die SPA soll einen Header haben, der euer Festivallogo, eine Überschrift und eine Unterüberschrift enthält (siehe Abbildung 1). Diese soll in der `Header.svelte` (siehe Abbildung 2) angelegt werden.
- Danach soll es Elemente geben, über die Besucher:innen ihre Erfahrungsberichte eingeben können. Für einen Bericht sollen folgende Daten erfasst werden:
  - Nutzername: `<input type='text'>`
  - Rating: `<input type='number'>`
  - Titel: `<input type='text'>`

  - Bericht: `<textarea>`

  Der Input der Nutzer:innen soll auf folgende Kriterien geprüft werden:

  - Das Sterne-Rating muss zwischen 1 und 5 sein.
  - Der Bericht hat eine maximale Zeichenlänge von 250.
  - Der Nutzername und der Titel sollen auf eine Zeichenlänge von 30 begrenzt sein.

  Wenn der "Bericht senden"-Button gedrückt wurde und die Angaben alle Kriterien erfüllen, soll der Erfahrungsbericht als JavaScript-Object in der `stores.js` (siehe Abbildung 2), nach dem Prinzip der Writable Stores, gespeichert werden.

- Unterhalb der Eingabefelder sollen die Berichte aus der `stores.js` angezeigt werden. Jeder Bericht soll in einer Card-Komponente (`Card.svelte`) dargestellt und in einem Grid-Layout mit zwei Spalten angeordnet werden. Um das Grid-Layout kümmert sich die `Reviews.svelte`. Diese lädt auch die Daten aus der `stores.js` und gibt sie an die einzelnen Cards als Props weiter.
- In jeder Card soll oben der Nutzername und die Sternebewertung nebeneinander stehen. Darunter der Titel und der Bericht. Wenn 5 Sterne vergeben werden, sollen diese in Gelb angezeigt werden. Falls weniger als 5 gegeben wurden, sollen alle restlichen Sterne in Grau dargestellt werden. (siehe Abbildung 1). Die Icons für die Sterne sind bereits im Template vorhanden. Die Cards sollen eine einheitliche und feste Größe haben, bei der auch Eingaben mit Maximallänge dargestellt werden können.
- Insgesamt sollen fünf Svelte-Komponenten verwendet werden. Diese sind in Abbildung 2 dargestellt.
- Damit die in der `stores.js` gespeicherten Informationen auch nach dem Schließen der Website erhalten bleiben, soll der `stores.js` mit dem `localStorage` des Browsers verknüpft werden. **Tipp:** Geht diese Anforderung erst nach der Abarbeitung aller anderen an.
- **Hinweis**: In Svelte 5 ist Reaktivität nicht mehr implizit wie in früheren Versionen. Lokale Zustände, die sich ändern und das UI beeinflussen sollen (z.B. beim Abspeichern von Berichten), müssen über spezielle Mechanismen wie `$state()` deklariert werden. Props, die an Komponenten übergeben werden, können mit `prop()` oder `$props()` eingebunden werden. Die automatische Aktualisierung des Interfaces funktioniert nur, wenn die genutzten Variablen korrekt reaktiv angelegt sind.

Anforderungen an Design und Layout

- Die Anwendung soll ein klares, übersichtliches und einheitliches Design besitzen. Dazu zählt bspw. ein stimmiges Farbschema.
- Die Anwendung soll ein klares, übersichtliches und einheitliches Layout besitzen. Dargestellte Inhalte sollen also bspw. immer übersichtlich angeordnet werden.

Wir wünschen Euch viel Erfolg
bei der Bearbeitung der Teilleistung!

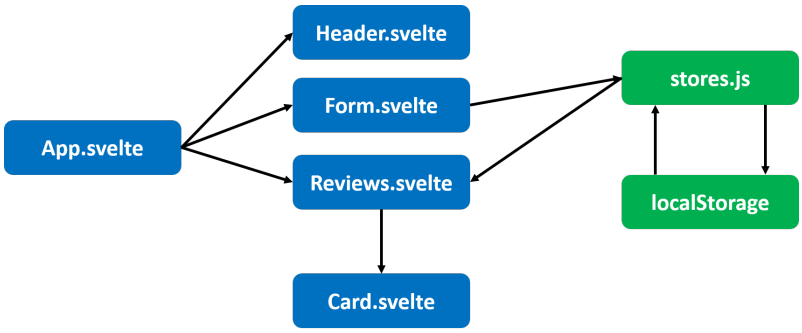

Abbildung 2: (Komponenten-)Struktur der Anwendung