Dokumentation Gästebuch

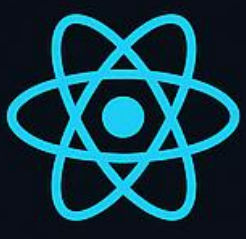
Gästebucheinträge

Anna
Tolle seite! Danke für die Arbeit

Ben
Ich freue mich auf mehr Features!
Loschen Bearbeiten

Clara
Sehr cooles Projeki, weiter so!
Loschen Bearbeiten

a
stydacs
Loschen Bearbeiten



Inhaltsverzeichnis

1. Projektidee

Zu Beginn der Projektplanung hatte ich die Idee, ein textbasiertes Fantasy-Adventure-Spiel zu entwickeln, das auf einem mehrstufigen Eventsystem basiert. In diesem Spiel sollten Spieler durch eine düstere Welt namens „Schattendorn“ reisen.

Diese Grundidee hatte ich bereits teilweise technisch im Backend umgesetzt, inklusive Datenmodell für Events, Eventstufen, Good/Bad-Tags, Optionen und Outcomes.

Im Laufe des Projekts habe ich jedoch gemerkt, dass der Umfang und die Komplexität dieser Spielmechanik den zeitlichen Rahmen des Moduls bei weitem überschreiten würden. Insbesondere die große Anzahl an Events und die damit verbundene Pflege der Datenbank hätten den Fokus stark vom eigentlichen Lernziel (Fullstack-Entwicklung mit REST-API) abgelenkt. Nach dieser Einsicht entschied ich mich, das Projekt deutlich zu vereinfachen und den Fokus wieder auf das Wesentliche zu legen: eine gut strukturierte Webapplikation mit klarer Funktionalität, sinnvoller Datenhaltung, validierter Eingabe und durchdachter Benutzerführung.

So entstand das GuestBook, eine minimalistische, aber funktionale Webapplikation, mit der Benutzer Gästebucheinträge erstellen, anzeigen, bearbeiten und löschen können. Die Kernmechanik der CRUD-Operationen bleibt erhalten, doch die Komplexität wurde deutlich reduziert. Interessanterweise profitierte das neue Projekt vom Vorprojekt: Einige Namenskonventionen oder bereits vorbereitete Back-End-Komponenten wurden übernommen oder angepasst. Das Resultat ist ein schlankes, verständliches System, das sowohl die Anforderungen an die Frontend- als auch die Backend-Module erfüllt, ohne sich in zu aufwendigen Spielsystemen zu verlieren.

2. Anforderungskatalog

- Benutzer können neue Gästebucheinträge erstellen
- Alle Einträge sollen als Liste angezeigt werden
- Einträge können editiert werden
- Einträge können gelöscht werden
- Das System verwendet eine relationale Datenbank zur Datenspeicherung
- Die REST-API validiert alle Eingaben serverseitig
- Die Benutzeroberfläche ist übersichtlich und reaktiv
- Die Anwendung enthält Unit-Tests für Front- und Backend

3. Use Cases & User Stories

Use Case 1 – Eintrag erstellen

- Als Benutzer
- möchte ich einen neuen Gästebucheintrag erfassen
- damit ich meine Nachricht öffentlich teilen kann.
- Akzeptanzkriterien: Formular validiert Eingaben, Eintrag erscheint in Liste.

Use Case 2 – Einträge anzeigen

- Als Besucher
- möchte ich alle bisherigen Einträge sehen
- damit ich Nachrichten anderer lesen kann.
- Akzeptanzkriterien: Liste mit vollständigen Einträgen.

Use Case 3 – Eintrag bearbeiten

- Als Benutzer
- möchte ich einen Eintrag nachträglich anpassen können
- damit ich Tippfehler oder Änderungen vornehmen kann.
- Akzeptanzkriterien: Nur gültige Einträge, Änderungen werden gespeichert.

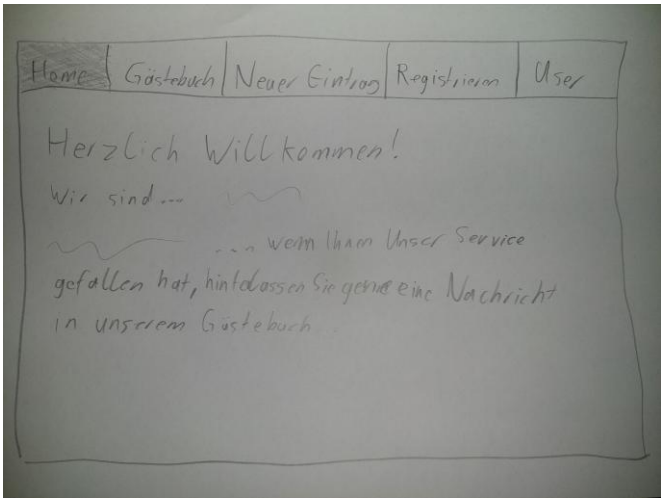
Use Case 4 – Eintrag löschen

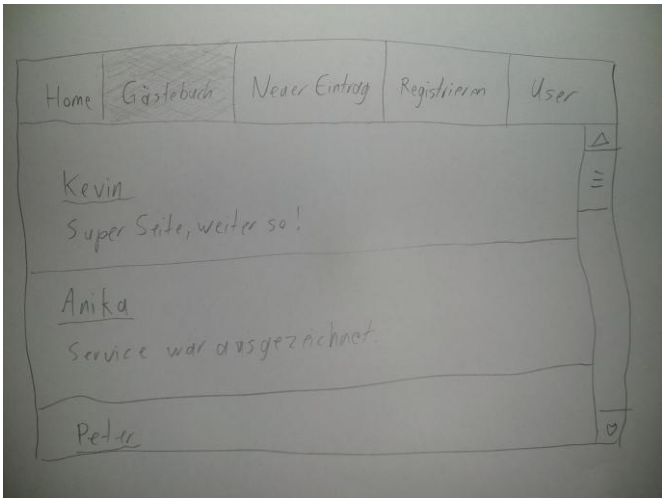
- Als Benutzer
- möchte ich einen Eintrag löschen
- damit dieser nicht mehr öffentlich sichtbar ist.
- Akzeptanzkriterien: Eintrag verschwindet.

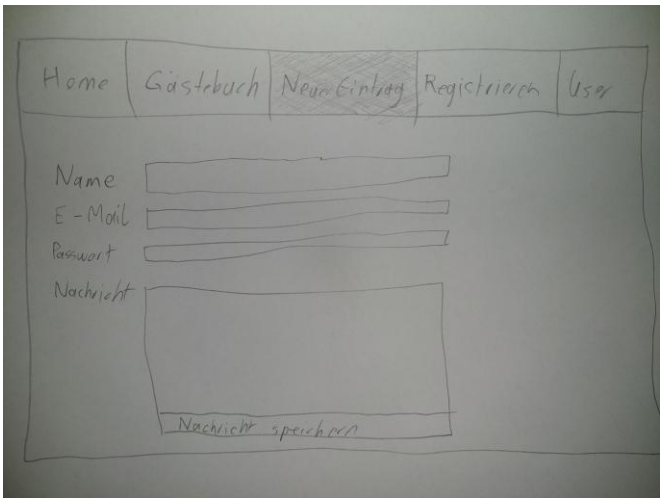
4. Zeitplanung

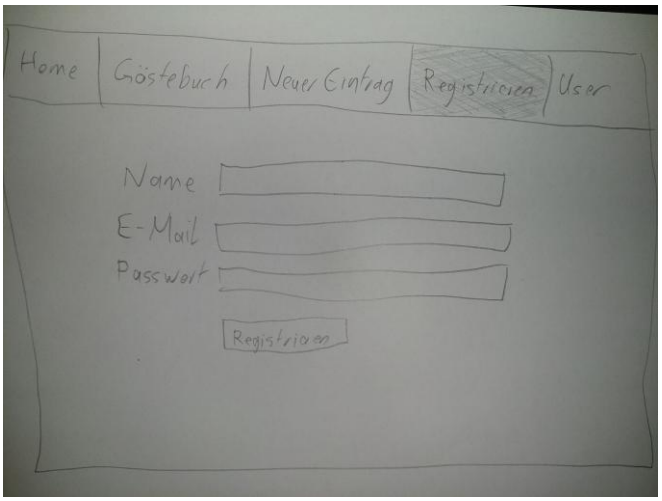
Aufgabe	Geschätzter Aufwand
Projektidee & Konzept	1h
Use Cases & Planung	2h
Backend-Datenmodell & API	4h
REST-API Implementierung	6h
Frontend-Komponenten React	6h
UI-Design & Routing	3h
Validierung & Error Handling	2h
Testing	3h
Dokumentation schreiben	4h
Total	**32h**

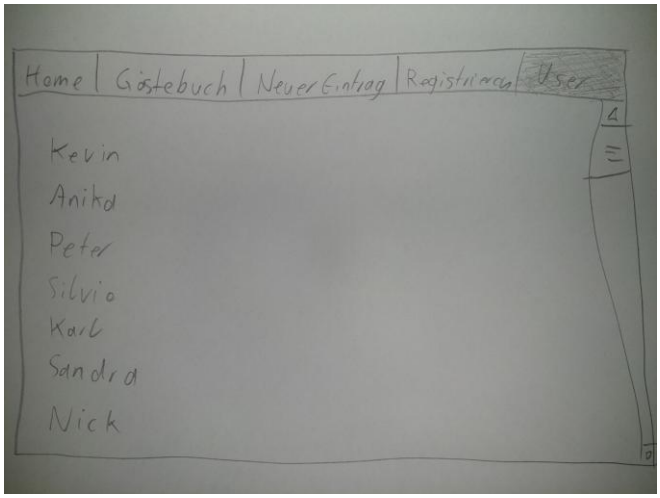
5. Storyboard / Mockups





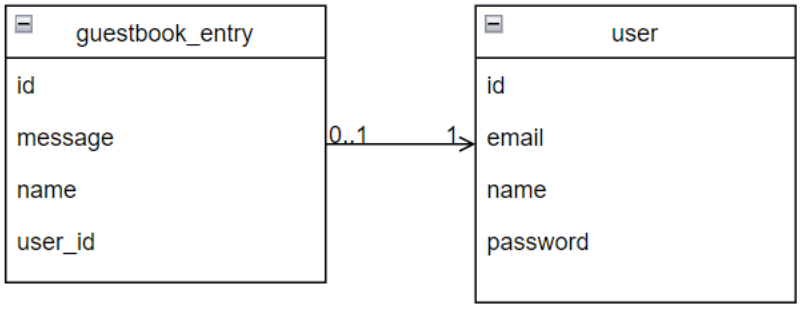






6. Klassendiagramm

Es gibt jeweils eine Klasse mit Gästebucheinträgen und eine mit Nutzern um die Datenbank zu erzeugen.



7. REST-Schnittstellen

Gästebuch-Endpunkte

HTTP	Pfad	Beschreibung	Beispiel-Request	Beispiel-Response
GET	/api/guestbook	Gibt alle Gästebucheinträge zurück	-	Liste von Einträgen
POST	/api/guestbook	Erstellt einen neuen Eintrag	{ "name": "Max", "message": "Hallo Welt" }	Eintrag mit ID
PUT	/api/guestbook/{id}	Aktualisiert einen Eintrag anhand der ID	{ "name": "Max", "message": "Update" }	Aktualisierter Eintrag
DELETE	/api/guestbook/{id}	Löscht den Eintrag mit der angegebenen ID	-	HTTP 204 No Content

Benutzer-Endpunkte

HTTP	Pfad	Beschreibung	Beispiel-Request	Beispiel-Response
GET	/api/users	Gibt alle Benutzer zurück	-	Liste von Benutzern
POST	/api/users	Erstellt einen neuen Benutzer	{ "name": "Max", "email": "max@mail.com", "password": "1234" }	Gespeicherter Benutzer
POST	/api/users/verify	Prüft Benutzer auf Übereinstimmung	{ "name": "Max", "email": "max@mail.com", "password": "1234" }	true oder false

Home-Endpunkte

HTTP	Pfad	Beschreibung	Beispiel-Request	Beispiel-Response
GET	/	Gibt eine einfache Textmeldung zurück, ob das Backend aktiv ist	-	Gästebuch Backend läuft!

8. Testplan & Testergebnisse

Backend Unit Tests

Nr.	Testbeschreibung	Eingabe	Erwartetes Verhalten	Testergebnis
1	Gästebucheintrag erfolgreich hinzufügen	Name = "Max", Message = "Hallo Welt"	Neuer Eintrag wird gespeichert und korrekt zurückgegeben	Bestanden
2	Alle Gästebucheinträge abrufen	-	Liste mit vorhandenen Einträgen (z. B. Alice, Bob) wird zurückgegeben	Bestanden
3	Eintrag löschen	ID = 1L	deleteById() wird mit korrekter ID aufgerufen	Bestanden
4	Eintrag erfolgreich aktualisieren	ID = 1L, neue Message = "Neu"	Eintrag wird aktualisiert und zurückgegeben	Bestanden
5	Update-Versuch bei nicht vorhandenem Eintrag	ID = 999L	Antwort mit HTTP 404 / Not Found	Bestanden

Frontend Unit Tests

Nr.	Beschreibung	Komponente	Eingabe / Zustand	Erwartetes Verhalten	Testergebnis
1	Gästebucheinträge aus API laden und anzeigen	Guestbook	API liefert 2 Einträge	Namen „Anna“ und „Tschüss“ erscheinen auf der Seite	Bestanden
2	Login + Gästebucheintrag absenden	GuestbookForm	Benutzer gibt Name, Nachricht, E-Mail, Passwort ein	Zwei fetch-Aufrufe: Verifizierung & POST an API	Bestanden
3	Keine Einträge vorhanden	GuestbookList	API liefert leere Liste	Text „Keine Einträge vorhanden.“ wird angezeigt	Bestanden
4	Benutzer erfolgreich über Formular erstellen	UserForm	Name, E-Mail, Passwort werden eingegeben	fetch POST wird korrekt mit Nutzerdaten an API ausgeführt	Bestanden
5	Benutzerliste korrekt anzeigen	UserList	Liste mit Benutzern Anna & Lukas	Beide Namen erscheinen in der gerenderten Liste	Bestanden

9. Installationsanleitung

Voraussetzungen:

- Java 17+
- Node.js & npm
- IDE (z. B. IntelliJ + VSCode)

MySQL Workbench

MySQL Workbench starten

Einloggen auf localhost:3306

Script: create_database.sql ausführen

Backend starten:

cd backend
mvn spring-boot:run

Script: example_data_insert.sql

Frontend starten:

cd guestbook-frontend
npm install
npm start

Die Anwendung ist danach erreichbar unter: http://localhost:5173/

Tests Laufen lassen:

cd guestbook-frontend
npm test

cd backend
mvn test

10. Hilfestellungen & Quellen

- ChatGPT (technische Rückfragen)
- StackOverflow
- React-Dokumentation
- Teams-Calls mit Mitschülern
- Bootstrap-Dokumentation
- Youtube

11. Fazit / Reflexion

Durch das Projekt konnte ich mein Wissen in der Fullstack-Webentwicklung deutlich vertiefen. Die Kombination von React im Frontend mit Spring Boot im Backend hat mir geholfen, den Gesamtprozess einer Webapplikation von der Planung bis zum Deployment zu verstehen. Besonders herausfordernd war die Konsistenz der Daten zwischen Frontend und Backend zu gewährleisten. Insgesamt bin ich mit dem Ergebnis sehr zufrieden.