

# Projekt Dokumentation: Event-Ticketing-Plattform API

## 1. Einleitung und Kernziel

Dieses Dokument beschreibt die Architektur, Technologieauswahl und die wichtigsten technischen Entscheidungen der entwickelten Event-Ticketing-Plattform API. Ziel ist es, ein Verständnis für die Struktur, Funktion und Integrität des Projekts zu vermitteln.

Produkt	Event-Ticketing-System zur Verwaltung von Theater, Konzerten und Seminaren.
Kernziel	Aufbau einer sicheren, skalierbaren REST API mit Fokus auf die Datenintegrität im Verkaufsprozess.
Status	Backend läuft stabil (Java/Spring Boot), Kommunikation mit Frontend hergestellt. Server läuft Stabil Tomcat läuft auf <a href="http://localhost:8080">http://localhost:8080</a> . Frontend Sichtbarkeit hergestellt, die Seite lädt lokal. Der CORS-Filter erlaubt Anfragen von Port 5500 an das Backend. Datenstruktur ist fehlerfrei, alle Entitäten (Models) sind korrekt definiert.

The screenshot displays the frontend interface of the Event-Ticketing Platform API, titled "Veranstaltung Köln". The interface features a header with the text "Erlebe die besten Events in Köln - Konzerte, Kultur & mehr!". Below the header, there are three main sections: "AC/DC" (a band image), "AC/DC" (a band image), and "AC/DC" (a band image). The interface includes a search bar, a category list (Rock, Jazz, Classic), an event list, an event calendar, a login/register section, and an admin section.

The bottom part of the screenshot shows the backend API response for a GET request to `http://localhost:8080/api/v1/events`. The response is a JSON array of event objects. The first event object is:

```
{
  "id": 3,
  "title": "[value 5]",
  "datum": null,
  "isPublished": true,
  "organisationsName": "Event GmbH",
  "locationName": "Lanxess Arena",
  "eventName": "Concert"
}
```

The response status is 200 OK, and the response body is 735 B.

## 2. Technologie-Stack (Das Fundament des Projekts)

Komponente	Technologie	Erklärung
Die Basis Sprache	Java 21 (LTS)	Java ist die Basis-Sprache des Backends. Wir haben diese Sprache gewählt, weil sie der Industriestandard für <b>geschäftskritische, hochskalierbare Systeme</b> ist (z.B. Banken, große Plattformen). LTS = Long-Term Support.
Der Bauleiter (Anwendungs-Motor)	Spring Boot 3	Seine Hauptaufgabe ist es, die Architektur <b>vollautomatisch zusammenzusetzen</b> , damit sich die Entwickler auf die Geschäftslogik konzentrieren können. Es verbindet die drei logischen Schichten: „Controller (Eingang), Service (Logik), Repository (Datenbank)“.
Das Lagerhaus (die Daten)	MariaDB 10.x (ACID)	Der Ort, an dem die Daten dauerhaft und sicher gespeichert werden (die Datenbank). <b>ACID</b> ist eine Qualitätsgarantie: Es stellt sicher, dass Ticketverkäufe immer <b>zuverlässig und konsistent</b> ablaufen (keine Überbuchung).
Der Übersetzer (ORM)  	JPA/ Hibernate	Der "Übersetzer" zwischen der <b>Java-Sprache</b> (Objekten) und der <b>Datenbank-Sprache</b> (SQL-Tabellen). Wir können Code schreiben, ohne komplexe SQL-Befehle selbst tippen zu müssen. Spezifikationen: „die formalen Regeln, wie die Java-Objekte mit der Datenbank zusammenarbeiten sollen.“
Der Bauwerkzeug	Maven	Das automatische Werkzeug, das alle Bibliotheken verwaltet, den Code baut und die Anwendung startet.
Tooling	CORS-Filter	Kommunikation zwischen Frontend (:5500) und Backend (:8080)
Frontend Logik	HTML	Der Rahmen und Inhalt: Definiert die Struktur und den

		Inhalt der Webseite (Überschriften, Bilder, Formulare). Es ist das "Skelett" der Anwendung, das festlegt, wo die Event-Titel und die Ticket-Auswahlfelder platziert sind.
	CSS	<b>Das Aussehen:</b> Definiert das visuelle Design, die Farben, Schriftarten und das Layout (z.B. wie die Event-Karten angeordnet sind oder wie die Ticket-Auswahl formatiert ist).
	JavaScript (JS)	<b>Die Intelligenz und Verbindung: Das "Gehirn" der Webseite. JavaScript ist verantwortlich für die gesamte Dynamik der Seite.</b> <b>API-Aufrufe:</b> Sendet die Anfragen (GET, POST) an Ihr Java/Spring Boot Backend. <b>Datenverarbeitung:</b> Liest die JSON-Antworten des Backends (z.B. das EventDetailsDto). <b>DOM-Manipulation:</b> Aktualisiert die Webseite in Echtzeit (z.B. zeigt die berechnete verfügbar-Zahl an, ohne die Seite neu laden zu müssen).

### 3. Datenstruktur und Architektur

Das System basiert auf einem relationalen Datenmodell (RDM), das die folgenden Entitäten umfasst:

- TicketKategorie: Definiert die maximale kontingentMax (wie viele Tickets maximal existieren).
- Bestellung / Bestellposition – Speichert die Menge der tatsächlich verkauften Tickets.



## 4. Der Nächste Meilenstein: Transaktionssicherheit

Der wichtigste offene Punkt ist die Implementierung des sicheren Ticketverkaufs über den POST /orders-Endpunkt.

### 4.1. Die Magie von @Transactional

Die Logik des Ticketverkaufs wird in der Methode processTicketOrder() (im OrderService.java) mit der Annotation @Transactional versehen.

<b>Funktionalität</b>	<b>Sicherheitsgarantie (Wichtigkeit)</b>
Start des Kaufs	@Transactional erstellt einen unsichtbaren, unteilbaren Block.
Kapazitätsprüfung	Wenn die gekaufte Menge des Kontigent übersteigt...
Rollback Mechanismus	...wird ein Fehler ausgelöst und die gesamte Operation automatisch rückgängig gemacht (ROLLBACK). Kein unvollständiger Verkauf wird gespeichert!