Projekt: **Event-Tool**

Designbeschreibung – Version 1.0

Abgabedatum: 21.04.2025

Projektgruppe: BitWorks
Ansprechpartner: Tobias Schuhmacher – *Projektleiter*

1 Inhaltsverzeichnis

1. Allgemeines

Bei dem vorliegenden Projekt handelt es sich um die Entwicklung einer webbasierten Plattform mit dem Namen **„Event-Tool“**, die Organisationen (z. B. Unternehmen oder Vereinen) bei der Planung, Verwaltung und Durchführung von Veranstaltungen unterstützt. Das Projekt wird im Rahmen der Lehrveranstaltung *Software Engineering I* im 4. Semester durchgeführt.

Die Plattform verfolgt das Ziel, wiederkehrende Abläufe wie die Teilnehmerverwaltung, die interne Kommunikation oder den Dokumentenaustausch bei Events zu automatisieren. Hierfür können Events mit individuellen Prozessabläufen erstellt werden, bei denen z. B. automatische E-Mails, Statusänderungen oder Upload-Bereiche integriert sind.

Das Projektteam setzt die Lösung als Fullstack-Webanwendung mit .NET-Technologien um. Das Frontend wird mit **Blazor WebAssembly**, das Backend mit **ASP.NET Core Web API** realisiert. Die Daten werden in einer **PostgreSQL-Datenbank** gespeichert. Als Programmiersprache wird **C#** verwendet. Die Authentifizierung erfolgt über ein **JWT-basiertes Login-System**. Die Entwicklung erfolgt kollaborativ über **GitHub** unter Einhaltung einer klaren Projektstruktur.

2. Produktübersicht

Die Plattform bietet registrierten Benutzern die Möglichkeit, sich über Events ihrer Organisation zu informieren, an diesen teilzunehmen und über automatisierte Prozesse mit den Organisatoren zu interagieren. Eine Organisation wird durch ihre E-Mail-Domain eindeutig identifiziert und kann eigene Events unabhängig verwalten.

Die Benutzer der Plattform nehmen verschiedene Rollen ein:

- **Mitglieder** können sich für Events anmelden und eigene Daten verwalten.
- **Organisatoren** erstellen und verwalten Events und Prozesse innerhalb ihrer Organisation.
- **Owner** besitzen darüber hinaus administrative Rechte für die Organisation.
- **Administratoren** verwalten Organisationen systemweit.

Ein zentrales Feature der Plattform ist die **Event-Prozesssteuerung**: Organisatoren definieren Prozessvorlagen, die z. B. automatische Erinnerungen, Uploadbereiche oder Statusänderungen enthalten. Diese Prozesse werden durch sogenannte Triggerpunkte (Datum, Teilnehmerzahl, etc.) ausgelöst.

Darüber hinaus bietet das System Funktionen wie:

- Event-Vorlagen zur Wiederverwendung
- Up- und Downloadbereiche für Eventunterlagen
- Teilnehmerverwaltung mit manueller und automatischer Steuerung

- Einladungen für externe Personen
- Q&A-Funktion für große Events

Die Benutzeroberfläche ist responsiv für Desktop- und Tablet-Geräte ausgelegt und besteht aus verschiedenen **Views**, z. B. Home-View, Event-View, Mitglieder-View oder Prozess-View.

3. Grundsätzliche Struktur- und Entwurfsentscheidungen

In diesem Abschnitt werden die grundlegenden Architekturentscheidungen, die verwendeten Technologien sowie zentrale Entwurfsmuster und Prinzipien vorgestellt. Ziel ist es, die interne Struktur des Systems so darzustellen, dass neue Entwickler:innen die Umsetzung nachvollziehen und sich effizient in die Codebasis einarbeiten können.

3.1. Verwendete Architektur

Die Anwendung ist als **mehrschichtige Client-Server-Architektur** umgesetzt. Die einzelnen Schichten sind klar voneinander getrennt:

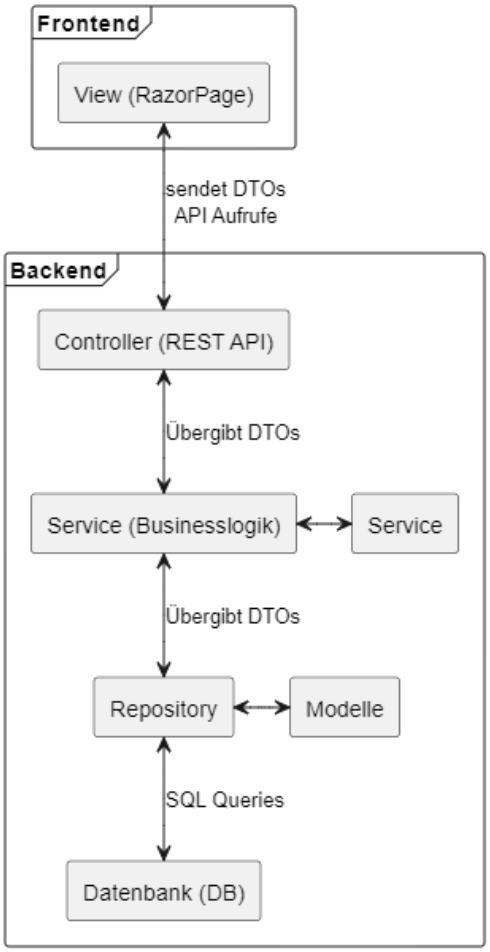
- **Frontend (Blazor WebAssembly):** Präsentation und Benutzerinteraktion im Browser. Kommuniziert per HTTP mit dem Backend (REST-API).
- **Controller-Schicht (ASP.NET Core):** Stellt REST-Endpunkte bereit, nimmt Anfragen entgegen, validiert Daten und ruft interne Services auf.
- **Service-Schicht:** Enthält die vollständige **Businesslogik**, z. B. Rollenverwaltung, Validierung von Teilnehmerzahlen oder Verarbeitung von Prozess-Triggern.
- **Persistenz-Schicht (Repositories):** Verwaltet alle Datenzugriffe auf die PostgreSQL-Datenbank über **Dapper**. Die Repositories kapseln SQL-Abfragen und bieten eine saubere Schnittstelle zur Datenbank.

Diese Struktur erlaubt eine hohe Modularität, gute Testbarkeit und klare Verantwortlichkeiten. Änderungen in der Datenbank oder im Frontend beeinträchtigen nicht die Logik in anderen Schichten.

3.1.1. Schichtendiagramm

Zur besseren Veranschaulichung der internen Architektur des Event-Tools wurde ein vereinfachtes Schichtendiagramm erstellt. Es stellt die grundlegende Architektur der einzelnen Schichten dar.
Im Folgenden werden die Funktionen der einzelnen Schichten näher erläutert.

View (RazorPage): Von hier werden die API-Aufrufe gesendet. Führt der Nutzer eine Aktion aus, werden hier die Befehle gesendet. Die zu übermittelnden Informationen werden hier über DTOs mitgegeben.
Controller (REST API): Hier werden die Befehle und Aktionen betrachtet und an den jeweiligen Service weitergegeben. Die Übergabe an die Services wird wieder durch DTOs gehandhabt.
Service (Businesslogik): Hier wird die tatsächliche Logik durchgeführt. Mithilfe der Übergebenen DTOs werden hier die einzelnen Befehle mithilfe konkreter Methoden durchgeführt. Dabei sind die einzelnen Services nach Verantwortlichkeiten gekapselt. Die Services können dabei auch untereinander kommunizieren. Wurden Änderungen an Objekten vorgenommen, werden diese mithilfe weiterer DTOs an die Repository weitergegeben.
Repository: Hier werden die erhaltenen DTOs verarbeitet und als tatsächliche SQL-Befehle an die Datenbank gegeben. Zusätzlich werden hier die Models der DB erstellt.
Datenbank: Hier werden die Informationen der Modele gespeichert. Mithilfe von SQL-Befehlen werden hier Daten erstellt, bearbeitet und gelöscht.



3.2. Verwendete Technologien und Tools

Komponente	**Technologie / Tool**
Programmiersprache	C# (.NET 8.0)
Frontend	Blazor WebAssembly
Backend	ASP.NET Core Web API
Datenbank	PostgreSQL
ORM	Dapper
Authentifizierung	JWT (JSON Web Tokens)
Tests	MSTest, xUnit
Versionskontrolle	Git (GitHub)
Projektmanagement	ClickUp
UML	PlantUML

3.3. Begründung der Architekturentscheidungen

Die Verwendung einer klar getrennten Architektur folgt den Prinzipien von Clean Code, Trennung der Verantwortlichkeiten und SOLID. Durch die Aufteilung in Services, Controllern und Repositories wird die Anwendung wartbar, gut testbar und erweiterbar.
Die Entscheidung für **REST** als Schnittstellentechnologie fiel zugunsten der Teamdurchführung: Im Gegensatz zu GraphQL bietet REST eine geringere Einstiegshürde, da keine zusätzliche Komplexität durch Query-Syntax, Resolver-Logik und Schema-Generierung entsteht. Für die Anforderungen des Projekts ist REST vollkommen ausreichend.
Durch den Einsatz von **JWT** kann die Rolle und Identität eines Benutzers bei jedem API-Zugriff sicher validiert werden. Dies ist besonders wichtig für die granulare Rollenverwaltung innerhalb von Organisationen.

3.4. Einsatz von DTOs (Data Transfer Objects)

Zur Trennung von internen Datenmodellen und API-Schnittstellen werden **DTOs** eingesetzt. Dies ermöglicht:

- die gezielte Freigabe von Daten nach außen,
- die Kapselung interner Logik (z. B. keine Passwörter im Response),
- die Validierung eingehender Daten unabhängig vom internen Modell,
- und eine langfristige Flexibilität bei der Weiterentwicklung der Businesslogik.

Die Zuordnung zwischen Domain-Modellen und DTOs erfolgt derzeit manuell, kann jedoch bei Bedarf später automatisiert werden (z. B. mit Mapster).

4. Struktur- und Entwurfsentscheidungen auf Paket- und Objektebene

Im Rahmen der objektorientierten Analyse (OOA) des Event-Tools wird die Systemstruktur sowohl auf statischer als auch auf dynamischer Ebene beschrieben. Der Fokus dieser Arbeit liegt jedoch auf der dynamischen Analyse, da die Plattform stark interaktive und zustandsbasierte Prozesse abbildet.

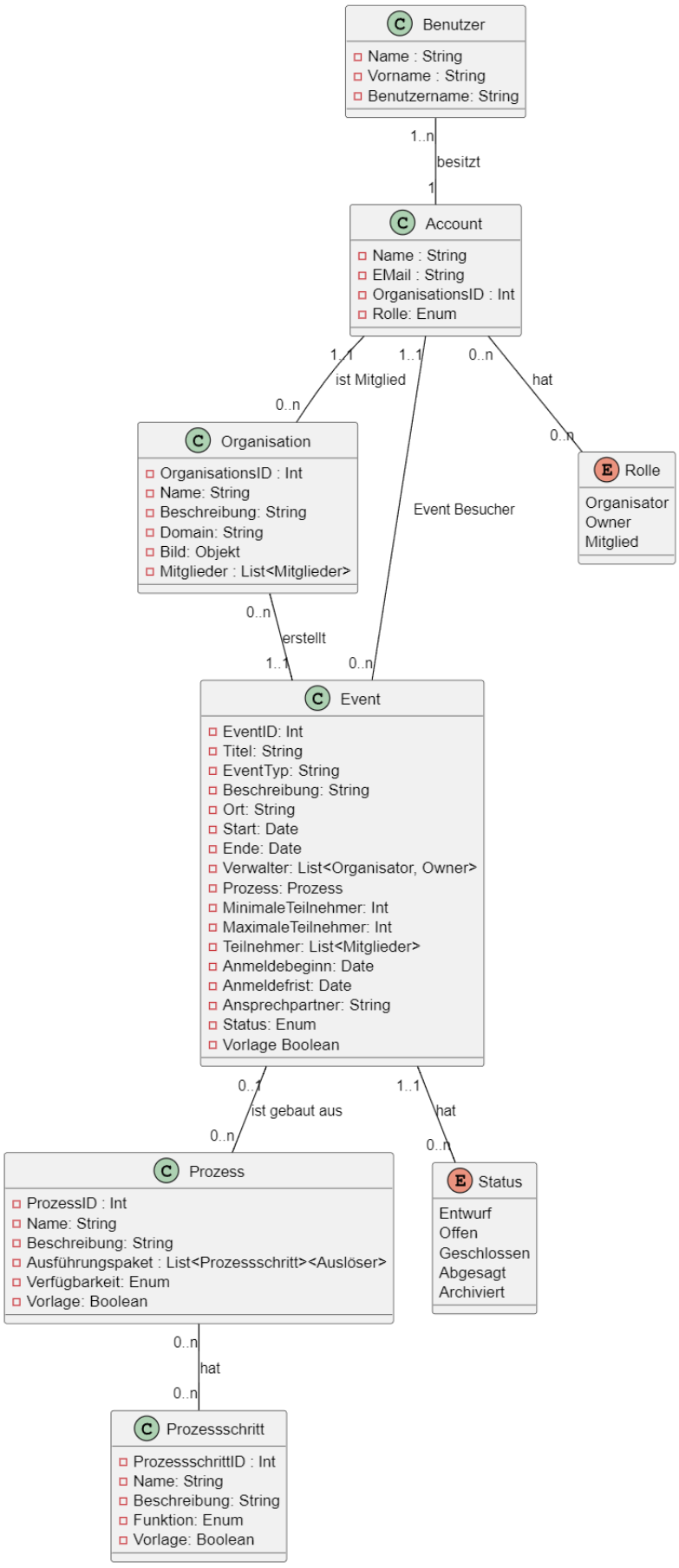
4.1. Statische Analyse

Die statische Analyse, bestehend aus Paket- und Klassendiagrammen, stellt die grundsätzlichen Strukturen und Beziehungen der Systemkomponenten dar.

4.1.1. Domänenmodell

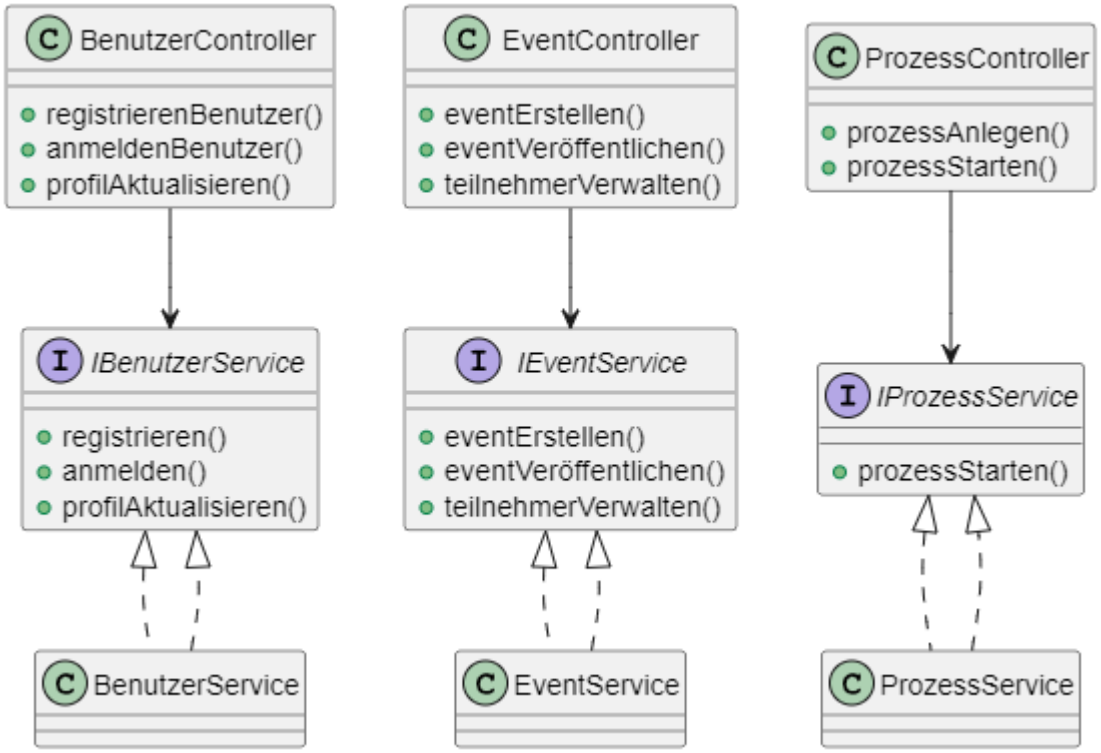
Das Domänenmodell stellt die zentralen Geschäftsobjekte des Systems dar. Hier werden die wichtigsten Entitäten und deren Verhalten abgebildet, die die Kernlogik der Event- und Organisationsverwaltung bestimmen.

Domänenmodell



4.1.2. Service-/Controller-Schicht

Die Services kapseln die Geschäftslogik des Systems und werden von den jeweiligen Controllern angesteuert, die als Schnittstelle zur Außenwelt (API-Schicht) dienen. Interfaces trennen die Definition von Verhalten und ermöglichen so eine flexible Weiterentwicklung.

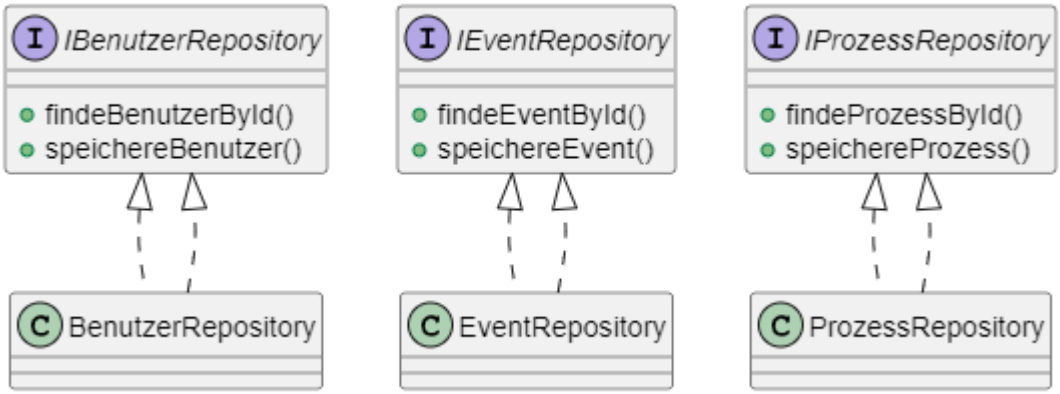


Es folgt eine Liste weiterer bereits vorhandener Services:

- AdministartionService
- OrganisationService
- PersonService
- AccountService
- DateiverwaltungsService
- SucheService
- AuthenticateService
- EmailService
- WorkerService
- LoggerService

4.1.3. Repository-Schicht

Die Repository-Schicht abstrahiert den Datenbankzugriff und kapselt CRUD-Operationen. Jede Repository-Klasse implementiert ein Interface, wodurch die Entkopplung von Logik und Persistenz gewährleistet wird.

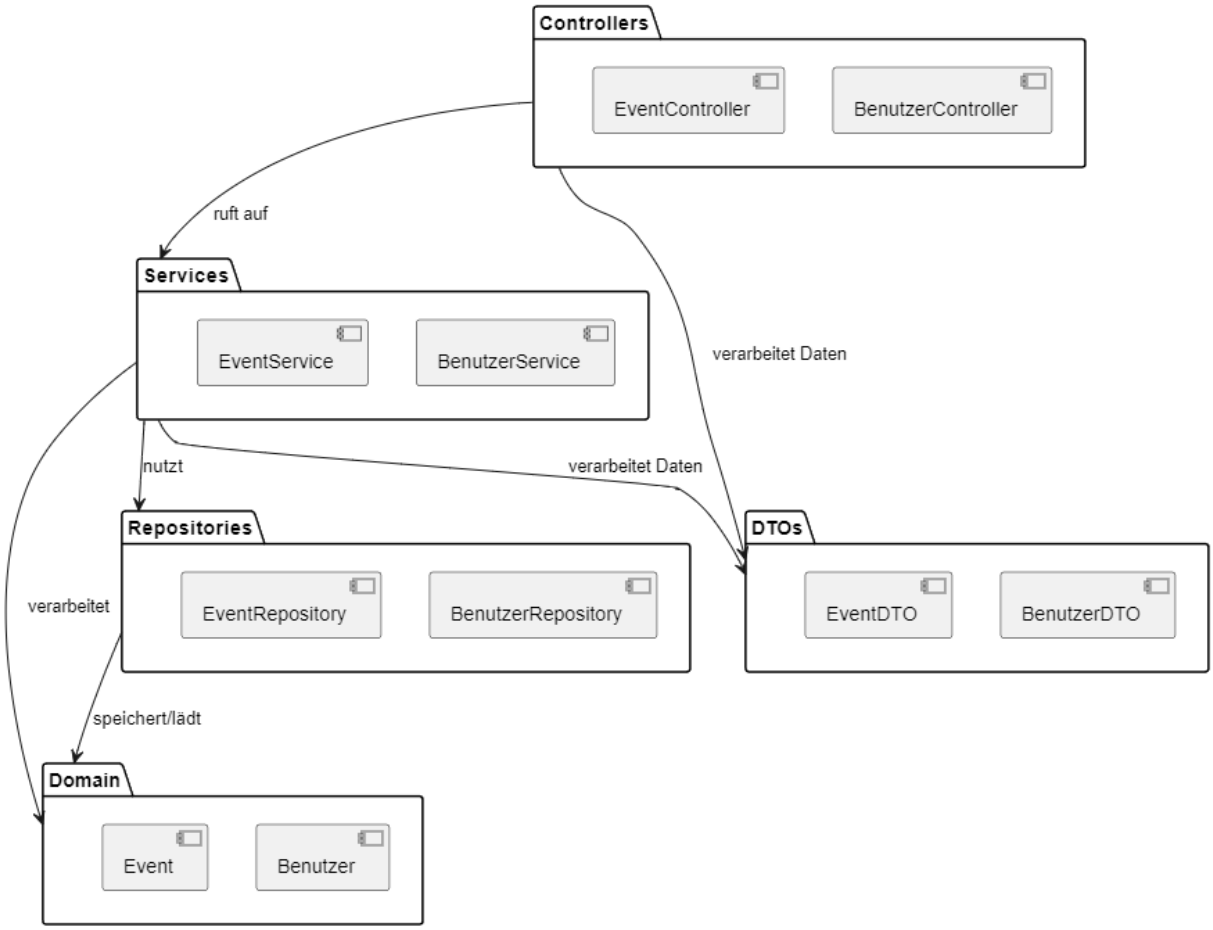


4.1.4. Pakete

Zur Veranschaulichung der logischen Struktur des Systems wurde ein Paketdiagramm erstellt. Es zeigt die wichtigsten Schichten (Pakete) und deren wesentliche Abhängigkeiten untereinander.

- Die **Controllers** bilden die API-Schnittstelle.
- Die **Services** kapseln die Anwendungslogik.
- Die **Repositories** sorgen für den Zugriff auf die Datenhaltung.
- Die **Domain**-Objekte repräsentieren die zentralen Geschäftsklassen.
- Die **DTOs** ermöglichen den Datenaustausch zwischen Client und Server.

Um die Übersichtlichkeit zu wahren, werden innerhalb der Pakete exemplarisch nur jeweils die zwei Klassen Benutzer und Event dargestellt.



4.2. Dynamische Analyse

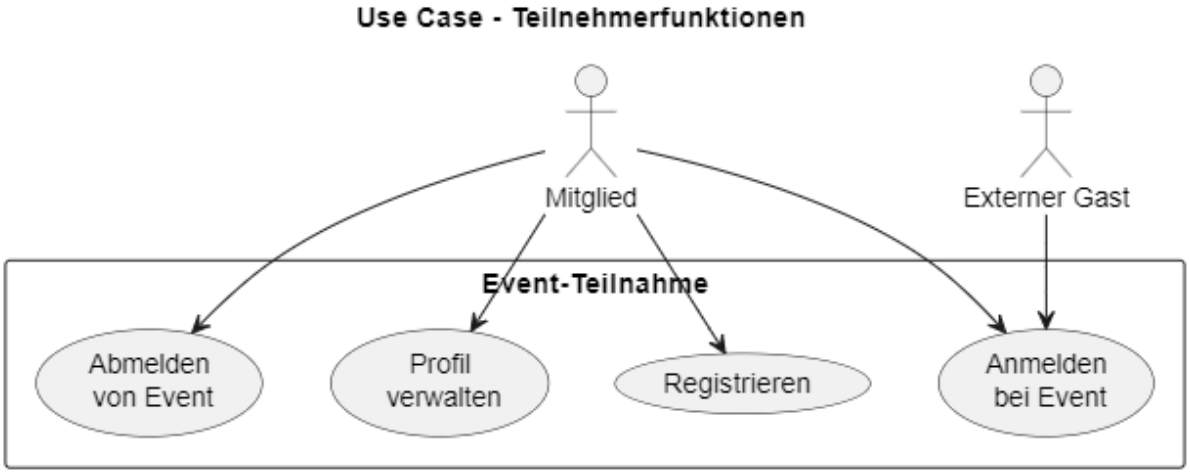
Ziel der dynamischen Analyse ist es, typische Anwendungsfälle und das Verhalten zentraler Systembestandteile in verschiedenen Szenarien zu modellieren. Hierzu werden Use-Case-Diagramme, Sequenzdiagramme sowie Zustandsdiagramme eingesetzt. Die Auswahl der dargestellten Anwendungsfälle orientiert sich an den funktionalen Kernprozessen des Systems.

4.2.1. Use-Cases

Die Use-Case-Diagramme zeigen die wichtigsten Benutzerrollen und deren Interaktionen mit dem System aus einer externen Perspektive. Dabei werden die Aufgaben der Teilnehmer, Organisatoren, Owner und Administratoren modelliert.

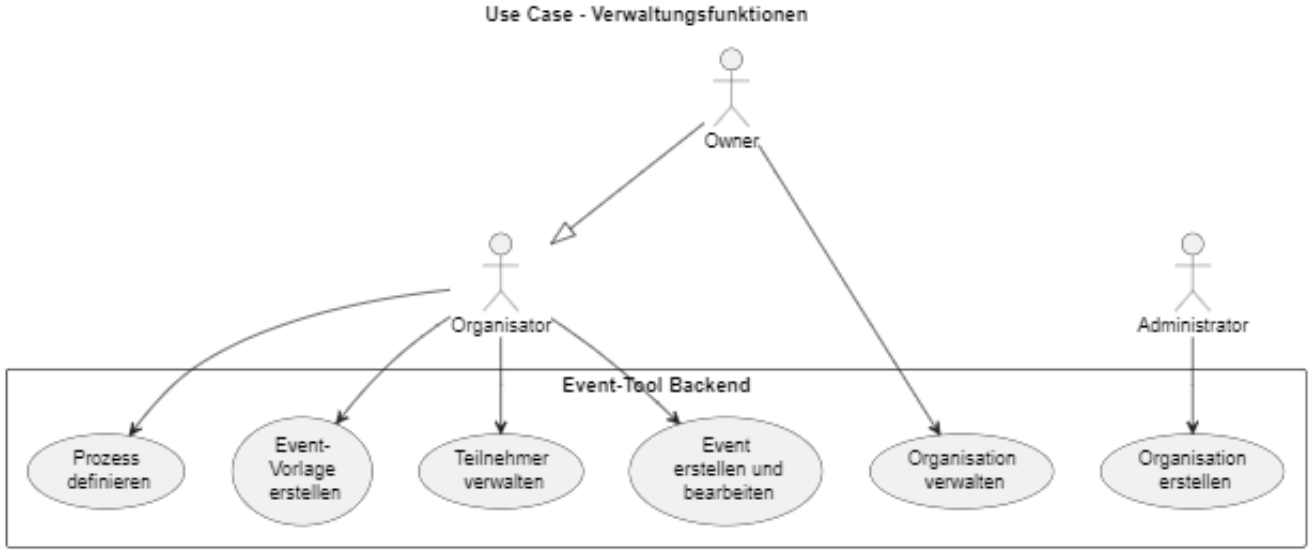
Teilnehmerperspektive

Dieses Use-Case-Diagramm beschreibt die Interaktionen von registrierten Mitgliedern und externen Gästen mit dem Event-Tool. Es verdeutlicht insbesondere die Kernfunktionalitäten für Endnutzer und bildet die Grundlage für typische Interaktionen im Tagesgeschäft.



Verwaltungsperspektive

Dieses Use-Case-Diagramm stellt die erweiterten Aufgaben der Verwaltungsrollen dar. Es zeigt die erweiterte Steuerungsmöglichkeit über die Plattform, insbesondere durch Organisatoren und Owner.



4.2.2. Sequenzen

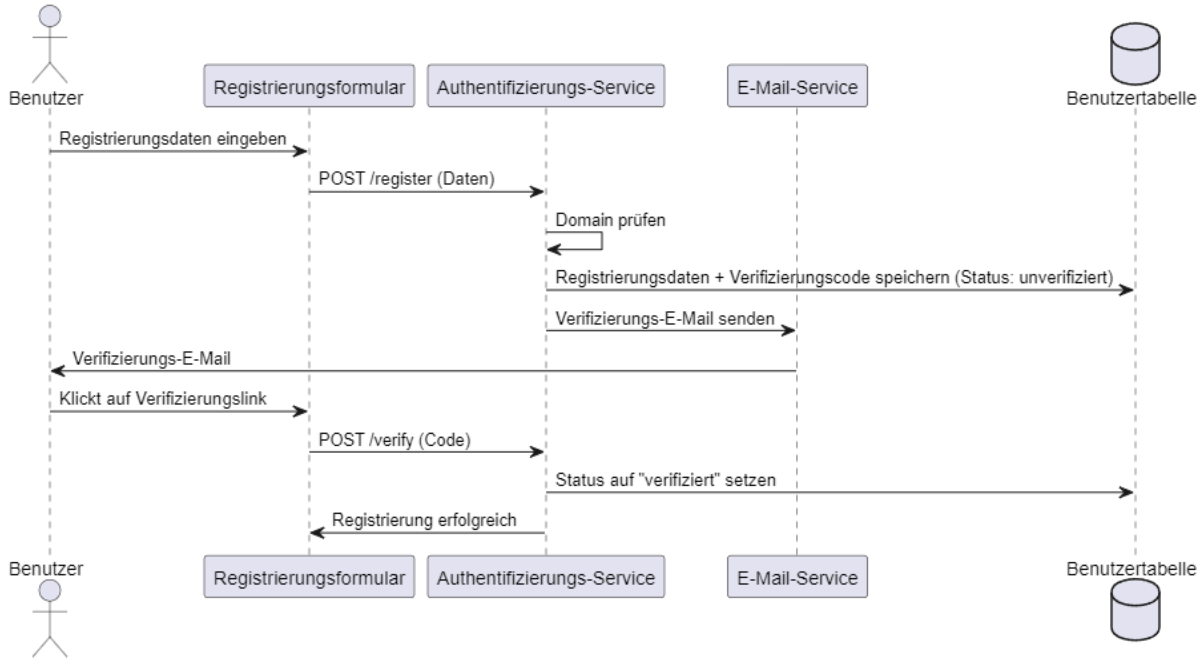
Die folgenden Sequenzdiagramme zeigen typische Interaktionen zwischen Benutzern und dem System im Ablauf einzelner wichtiger Geschäftsprozesse. Sie verdeutlichen die Kommunikation zwischen den beteiligten Objekten und die Reihenfolge der ausgetauschten Nachrichten.
Die Auswahl der Szenarien basiert auf zentralen Kernprozessen der Plattform sowie den im Testkonzept definierten TestzielenBitWorks_Testkonzept_v0….

Registrierung eines neuen Benutzers mit E-Mail-Verifikation

Die Registrierung eines neuen Benutzers stellt den Einstiegspunkt in das System dar. Neue Benutzer registrieren sich mit einer E-Mail-Adresse, die auf die Domain einer bestehenden Organisation verweist. Zur Sicherstellung der Gültigkeit wird eine Verifikations-E-Mail verschickt, die vom Benutzer bestätigt werden muss.

Ablaufbeschreibung:

1. Der Benutzer füllt das Registrierungsformular aus und sendet es ab.
2. Das Frontend übermittelt die Daten an den Authentifizierungs-Service.
3. Der Service prüft die E-Mail-Domain und erzeugt einen Verifizierungscode.
4. Der Verifizierungscode wird zusammen mit den Registrierungsdaten in der Datenbank gespeichert (Status: "unverifiziert").
5. Der E-Mail-Service versendet eine Bestätigungs-E-Mail an den Benutzer.
6. Der Benutzer klickt auf den Link in der E-Mail.
7. Das System validiert den Link, aktualisiert den Status des Benutzers auf "verifiziert" und ermöglicht den Login.

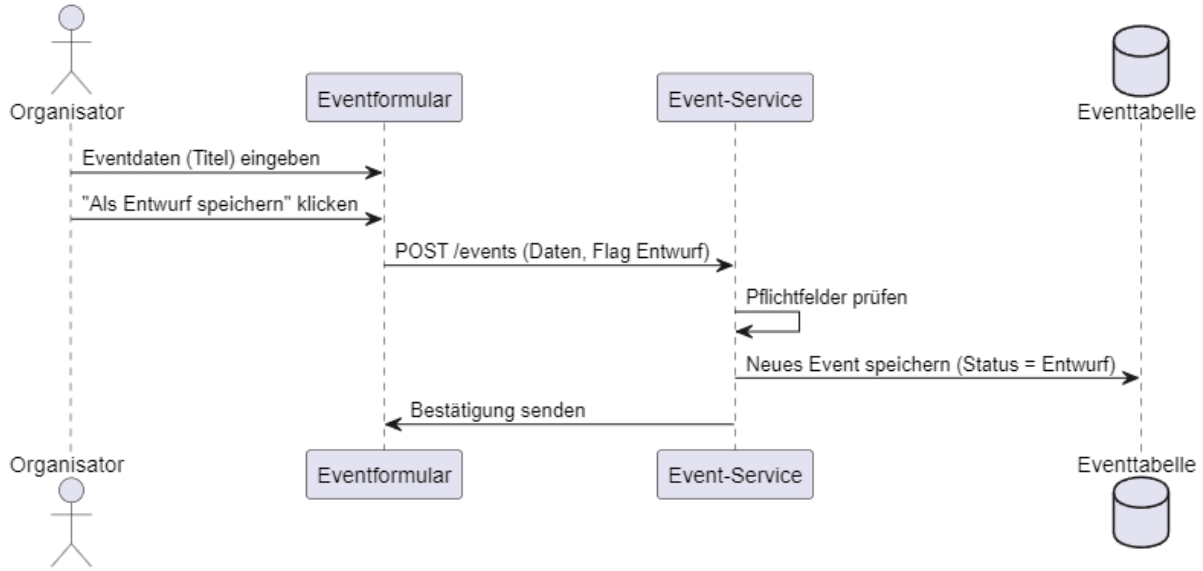


Erstellung eines neuen Events als Entwurf

Organisatoren oder Owner können neue Events anlegen. Dabei ist es möglich, das Event bereits während der Erstellung als **Entwurf** zu speichern, auch wenn nur das Pflichtfeld "Titel" ausgefüllt ist. Diese Funktion erlaubt ein späteres, schrittweises Ausfüllen der restlichen Eventdaten.

Ablaufbeschreibung:

1. Der Organisator/Owner öffnet das Eventformular und gibt erste Eventdaten (mindestens den Titel) ein.
2. Er entscheidet sich, das Event als Entwurf zu speichern.
3. Das Frontend sendet die eingegebenen Daten an den Event-Service.
4. Der Event-Service prüft die Daten und erkennt, dass nur ein Entwurf erstellt werden soll.
5. Ein neues Event wird in der Datenbank mit dem Status **"Entwurf"** gespeichert.
6. Der Organisator/Owner erhält eine Bestätigung und kann später das Event weiterbearbeiten.

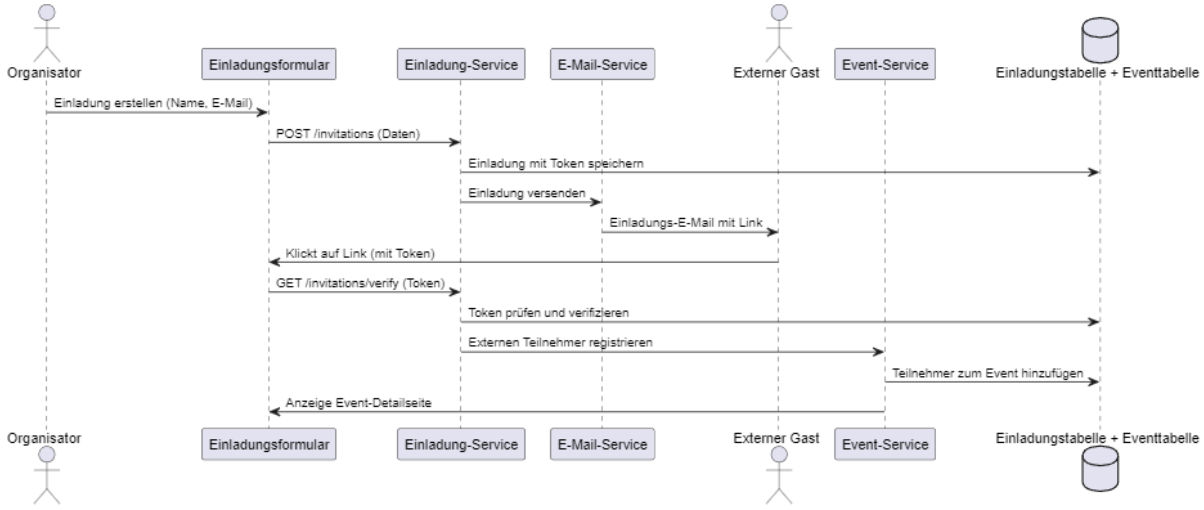


Einladung und Anmeldung eines externen Teilnehmers

Organisatoren oder Owner haben die Möglichkeit, externe Personen zu einem Event einzuladen, auch wenn diese noch keinen Account auf der Plattform besitzen. Der eingeladene Gast erhält eine personalisierte E-Mail mit einem Bestätigungslink. Über diesen Link kann sich der externe Teilnehmer direkt für das Event anmelden, ohne eine vollständige Registrierung durchlaufen zu müssen.

Ablaufbeschreibung:

1. Der Organisator/Owner öffnet das Einladungsformular und gibt Name sowie E-Mail-Adresse des externen Gastes ein.
2. Das Frontend übermittelt die Einladungsdaten an den Einladung-Service.
3. Der Einladung-Service erstellt eine Einladung mit einem eindeutigen Token in der Datenbank.
4. Der E-Mail-Service versendet eine Einladung an den externen Gast.
5. Der Gast klickt auf den Bestätigungslink in der E-Mail.
6. Das System prüft den Token und verknüpft den externen Gast mit dem Event als Teilnehmer.
7. Der Gast wird auf die Event-Detailseite geleitet und ist damit erfolgreich angemeldet.



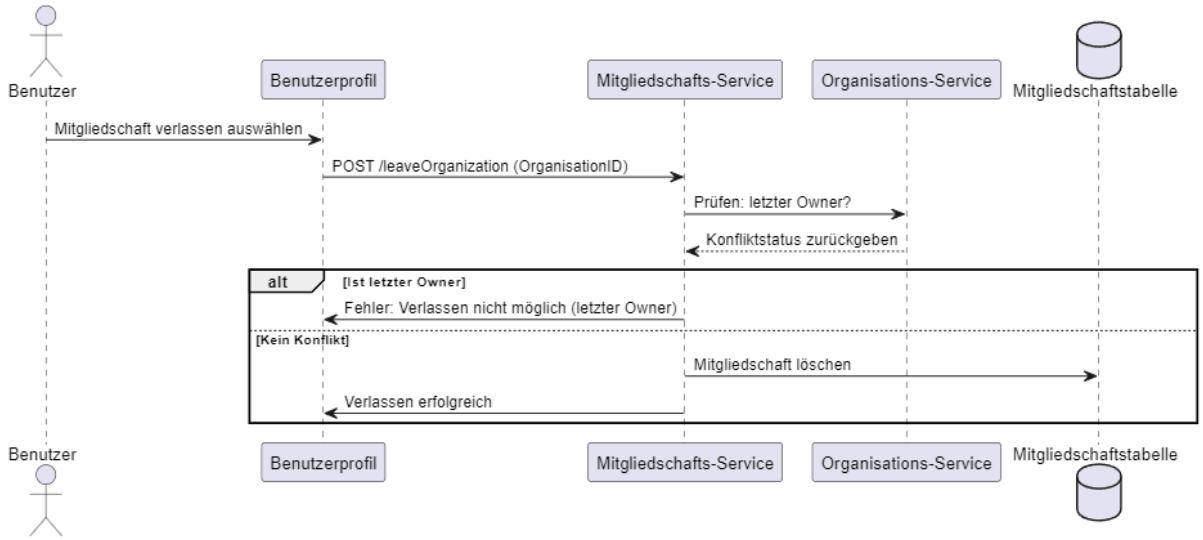
Benutzer verlässt eine Organisation

Benutzer haben die Möglichkeit, eine bestehende Organisation selbstständig zu verlassen. Dabei wird geprüft, ob der Benutzer der letzte Owner der Organisation ist.
Ein Benutzer darf die Organisation **nicht verlassen**, solange er die letzte Person mit Owner-Rechten ist, da jede Organisation mindestens einen Owner benötigt.
Event-Verwaltungen stellen hingegen keinen Hinderungsgrund dar: verlässt der Benutzer ein Event als letzter Verwalter, wird dieses Event intern als **"verwaist"** markiert.

Ablaufbeschreibung:

1. Der Benutzer öffnet seine Mitgliedschaftsübersicht im Benutzerprofil und wählt die Option, eine Organisation zu verlassen.
2. Das Frontend übermittelt die Anfrage an den Mitgliedschafts-Service.
3. Der Mitgliedschafts-Service fragt beim Organisations-Service an, ob der Benutzer der letzte Owner der Organisation ist.
4. Wird der Benutzer als letzter Owner erkannt, bricht der Mitgliedschafts-Service den Vorgang ab und informiert den Benutzer mit einer entsprechenden Fehlermeldung.
5. Liegt kein Konflikt vor, entfernt der Mitgliedschafts-Service die Mitgliedschaft des Benutzers aus der Datenbank.

6. Der Benutzer erhält eine Bestätigung über das erfolgreiche Verlassen der Organisation.



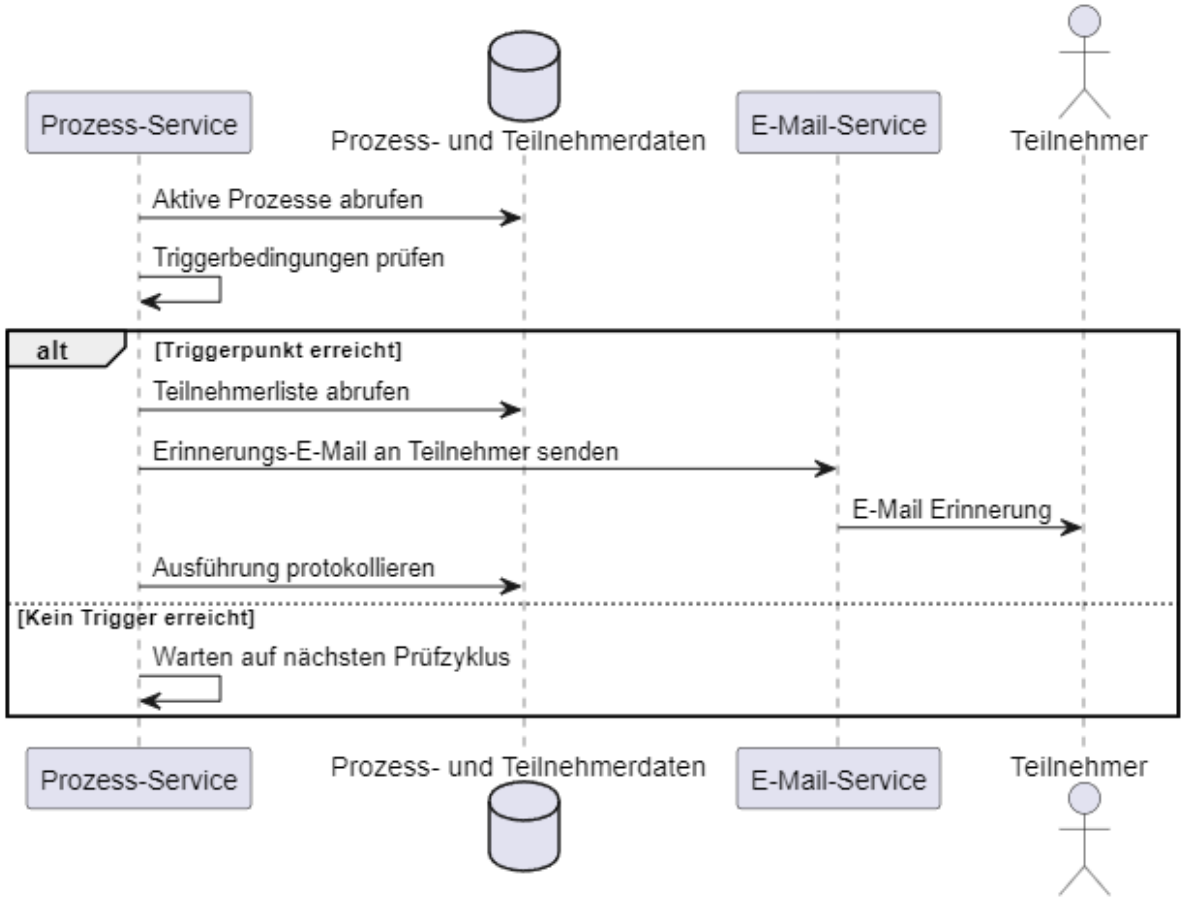
Automatischer Ablauf eines Prozessschritts bei einem Event

Innerhalb eines Events können durch definierte Prozessvorlagen bestimmte Aktionen automatisch ausgelöst werden. Diese Automatisierungen basieren auf sogenannten **Triggerpunkten** (z.B. Datum, Teilnehmerzahl erreicht).

Ein typisches Beispiel ist der automatische Versand einer Erinnerungs-E-Mail an alle Teilnehmer, wenn eine Woche vor dem Eventstart erreicht wird.

Ablaufbeschreibung:

1. Der Prozess-Service prüft in regelmäßigen Abständen (oder per Trigger) die definierten Prozessschritte der aktiven Events.
2. Beim Erreichen eines Triggerpunktes (z.B. ein bestimmtes Datum) wird der zugeordnete Prozessschritt automatisch ausgelöst.
3. Im Beispiel „E-Mail-Erinnerung“ wird eine E-Mail an alle Event-Teilnehmer vorbereitet.
4. Der E-Mail-Service übernimmt den Versand an alle betroffenen Teilnehmer.
5. Der Prozess-Service protokolliert die erfolgreiche Ausführung des Schrittes in der Datenbank.


Prozess-Service
Prozess- und Teilnehmerdaten
E-Mail-Service
Teilnehmer
Aktive Prozesse abrufen
Triggerbedingungen prüfen
alt
[Triggerpunkt erreicht]
Teilnehmerliste abrufen
Erinnerungs-E-Mail an Teilnehmer senden
E-Mail Erinnerung
Ausführung protokollieren
[Kein Trigger erreicht]
Warten auf nächsten Prüfzyklus
Prozess-Service
Prozess- und Teilnehmerdaten
E-Mail-Service
Teilnehmer

4.2.3. Zustände

Die Zustandsdiagramme zeigen, wie sich der Zustand eines Objektes im Laufe der Zeit in Abhängigkeit von Ereignissen verändert. Im Kontext des Event-Tools ist insbesondere der Lebenszyklus eines Events von zentraler Bedeutung, da Events verschiedene Status durchlaufen können, die jeweils unterschiedliche Sichtbarkeiten und Bearbeitungsmöglichkeiten zur Folge haben.

Typische Übergänge:

- Von **Entwurf** → **Offen**, wenn der Verwalter das Event manuell veröffentlicht (alle Pflichtfelder ausgefüllt).
- Von **Offen** → **Geschlossen**, wenn der Verwalter die Teilnahme schließt (z.B. weil Anmeldeschluss erreicht ist oder manuell).
- Von **Offen** → **Abgesagt**, wenn das Event aktiv abgesagt wird.
- Von **Geschlossen** → **Offen**, wenn die Teilnahme wieder geöffnet wird.
- Von **Geschlossen** → **Abgesagt**, wenn das Event abgesagt wird.
- Von **Abgesagt** → **Offen** oder **Geschlossen**, wenn das Event reaktiviert wird.
- Von **Geschlossen** → **Archiviert**, nach Ablauf des Events oder manuell durch Verwalter.
- Von **Abgesagt** → **Archiviert**, nach Ablauf des Events oder manuell durch Verwalter.

Die manuelle Statusänderung erfolgt über ein Dropdown-Menü im Eventverwaltungs-View, das nur für Eventverwalter verfügbar ist.