Designentwurf

Erstellt von:	Verifiziert von:
• **Sebastian Brosch**	• **Matti Bos**
• **Robin Schnaiter**	• **Dariusz Herzog**

Version	Erstellung	Beschreibung
1.0	21.04.2025	Erstellung Dokument
2.0	03.05.2025	Aktualisierung Dokument

Inhalt

Allgemein

Das Ziel ist es eine Online-Event-Management-Plattform im Auftrag der Organize-It GmbH zu entwickeln. Nach einer initialen Entwicklungsphase soll die Plattform von der Organize-It GmbH übernommen, weiterentwickelt und betrieben werden. Die Plattform ermöglicht es der Organize-It GmbH, interne physische Veranstaltungen für ihre Unternehmenskunden effizient zu planen, zu konfigurieren und zu verwalten. Der Fokus liegt auf der Benutzerverwaltung sowie der Automatisierung von Prozessen zur Automatisierung von Aufgaben bei der Verwaltung von Events. Nach Abschluss des Projekts wird die Plattform von Organize-It genutzt, um Events für verschiedene Unternehmen zu hosten.

Dieses Dokument beschreibt die wichtigsten Designentscheidungen, um eine Übersicht über das Projekt Turbo Events zu erhalten. Das Dokument richtet sich vor allem an Neueinsteiger des Projekts sowie als Referenz für die grundlegenden technischen und organisatorischen Strukturen des Projekts. Darüber hinaus dient das Dokument dazu, die Übergabe des Projekts nach Abschluss der initialen Entwicklungsphase an die Organize-It GmbH zu erleichtern.

Produktübersicht

Das folgende Diagramm zeigt die verschiedenen Akteure, Akteurinnen und Bereiche der Plattform und deren Beziehungen untereinander. Dies dient einer ersten Übersicht über das Gesamtsystem.

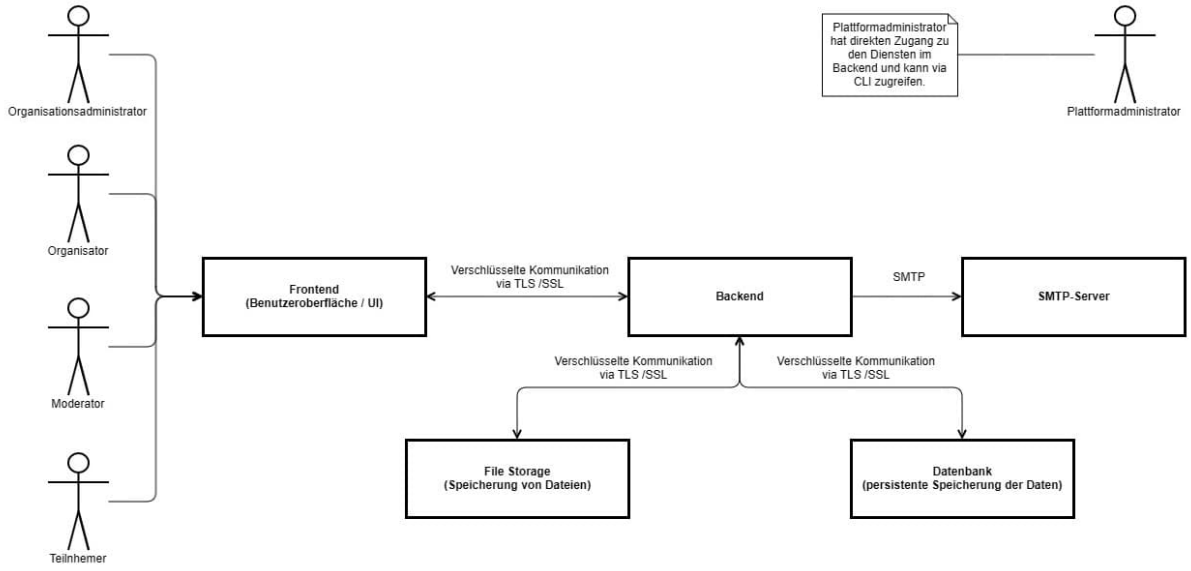

Abbildung 1: Übersicht der Akteure und grundlegenden Komponenten

Der zentrale Bereich der Anwendung ist das Backend. Dies dient als Vermittlungsschicht zwischen den Benutzer:innen und der Datenhaltung. Die Benutzer:innen können über das Frontend, und die dadurch geschaffene benutzerfreundliche Oberfläche alle Aktionen der Plattform bedienen. Die Oberfläche richtet sich dabei an Anwender:innen, welche technisch nicht versiert sind und dadurch intuitiv gestaltet werden muss. Über eine Verbindung zum Backend können Daten im Frontend angezeigt oder Daten aus dem Frontend in das Backend überführt werden. Die persistente Speicherung der Daten erfolgt über eine an das Backend

angebundene Datenbank. Die Dateien welche durch den Anwender:innen hochgeladen und auch wieder heruntergeladen werden können, werden in einem eigenen Datenspeicher gespeichert. Auch dieser ist über das Backend angebunden. Für den Versand von E-Mail-Nachrichten wird eine SMTP-Verbindung zu einem E-Mail-Server zur Verfügung gestellt. Sämtliche Kommunikation findet verschlüsselt statt, so dass Informationen auf dem Übertragungsweg nicht von Dritten gelesen werden können. Je nach Rolle haben Benutzer:innen Zugriff auf unterschiedliche Seiten und können dort spezifische Aktionen ausführen. Welche Rolle welche Seite sehen darf, zeigt das folgende Diagramm.

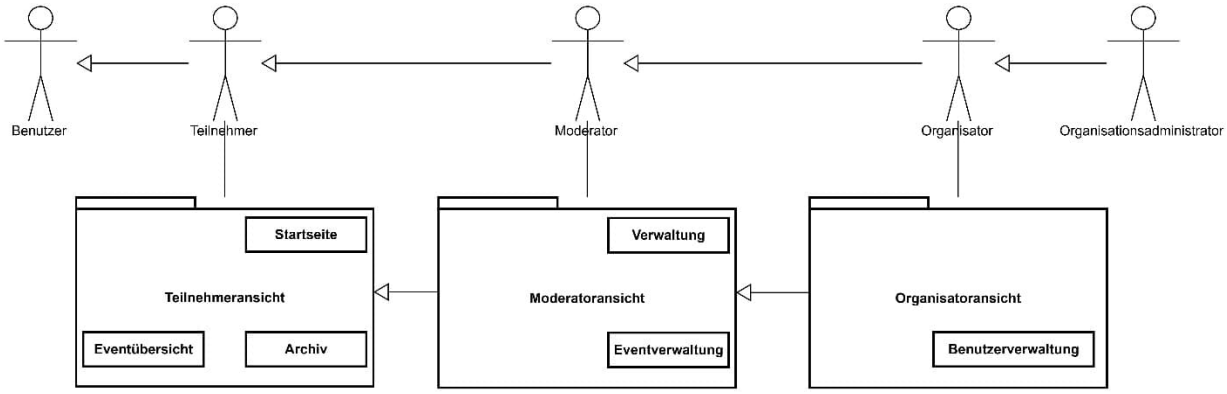

Abbildung 2: Akteure und deren Sichten

Wie bereits genannt ist ein wichtiger Teil der Software die Prozesssteuerung. Dabei gibt es zum einen die Prozessausführung und die Prozessverwaltung. Diese werden in den folgenden Aktivitätsdiagrammen gezeigt.

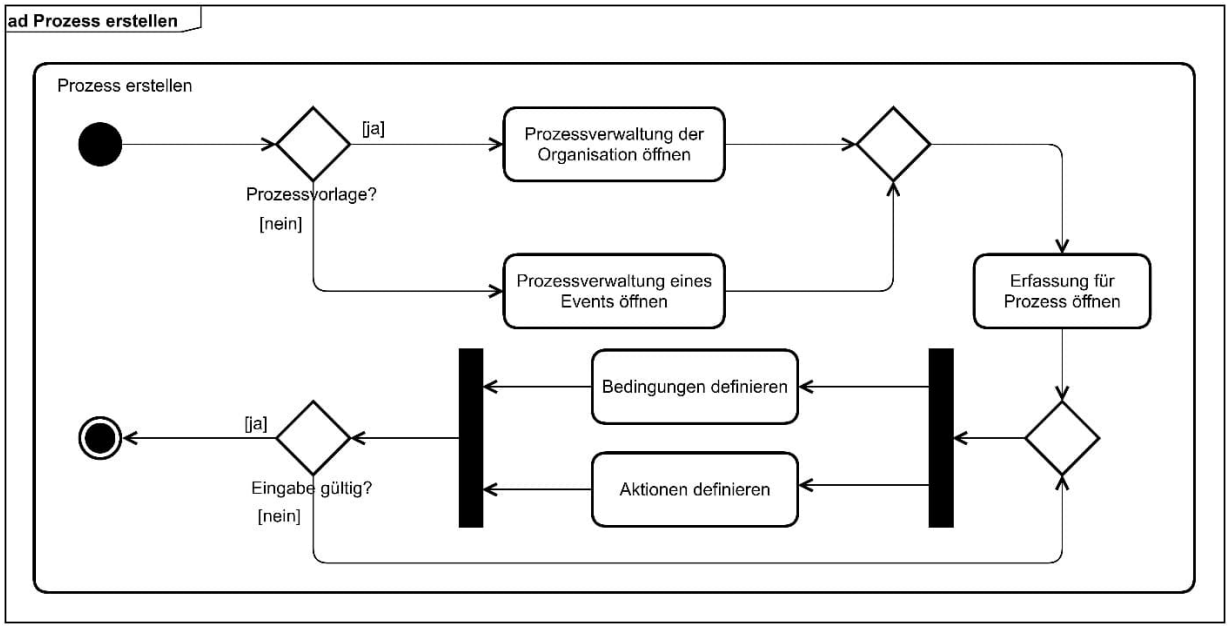

Abbildung 3: Erstellen eines Prozesses

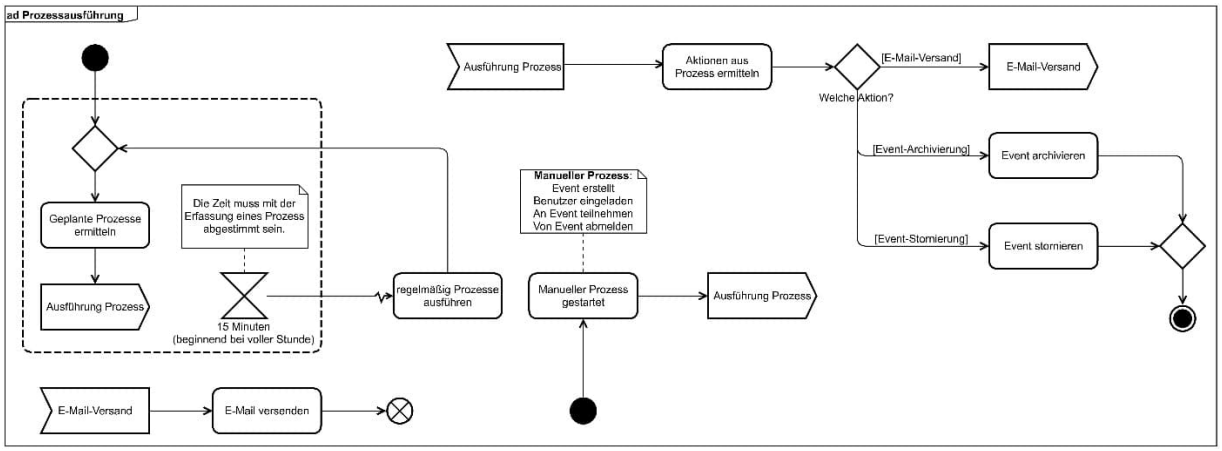

Abbildung 4: Ablauf der Prozessausführung

Struktur- und Entwurfsentscheidungen

Die Anwendung

Das folgende Komponentendiagramm zeigt erneut eine Übersicht der Plattform jedoch mit weiteren, auch technischen Details. In diesem Kapitel gehen wir auf die Gesamtstruktur der Anwendung ein.

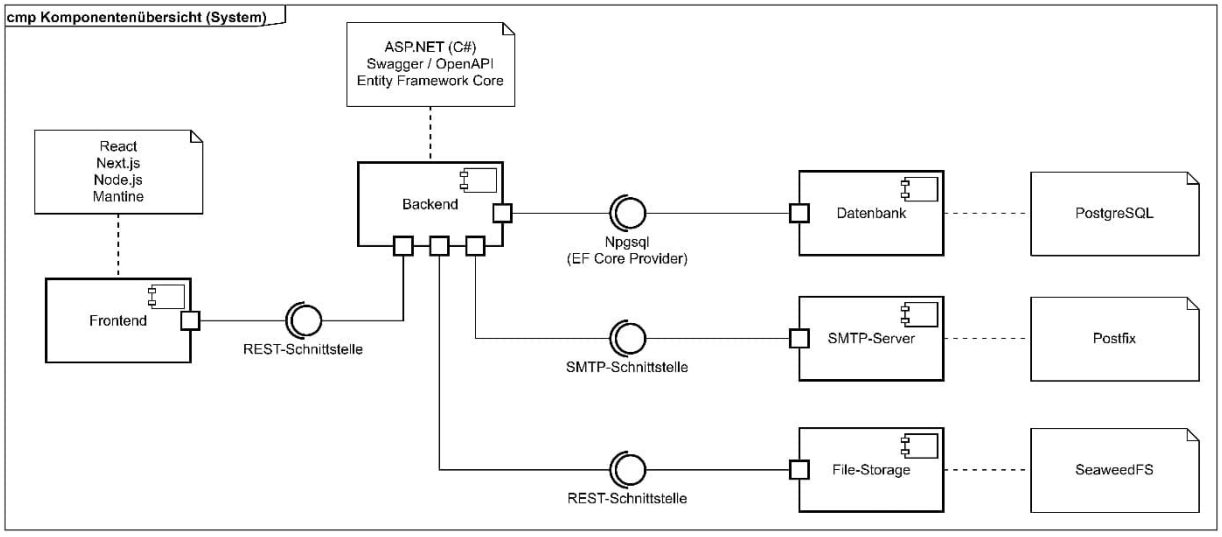

Abbildung 5: Komponentenübersicht des Systems

Die Anwendung wird im Client-Server-Modell aufgebaut. Der Austausch von Informationen zwischen den Clients und dem Server findet über eine RESTful-Schnittstelle statt. Im Frontend wird die grafische Benutzeroberfläche (UI) sowie die clientseitige Logik definiert und verwaltet. Im Backend wird die Geschäftslogik verarbeitet sowie die Datenhaltung vorgenommen. Das Backend stellt aber auch die RESTful-Schnittstelle zur Verfügung, so dass die Clients mit dem Backend kommunizieren können. Die Kommunikation des Clients mit der Anwendung sowie des Frontends mit dem Backend findet über HTTPS statt.

Die API wird mit Swagger dokumentiert, so dass Entwickler sich einen Überblick über die vorhandenen Endpunkte der API verschaffen können. Diese Endpunkte können auch als JSON-Datei aus Swagger exportiert und in Anwendungen wie Postman importiert werden.

Der Code wird als Clean Code entwickelt. Für die Umsetzung des Clean Code wurde eine Dokumentationskonzept erstellt, welches beispielsweise regelt, dass aussagekräftige Namen verwendet werden sollen. Aber auch die Größe von Funktionen soll so atomar wie möglich gehalten werden, so dass keine umfangreichen, hybriden Funktionen entstehen. Das soll Komplexität und Over-Engineering verhindern und die Wartung der Software auf lange Sicht leichter machen und neuen Entwicklern einen schnellen Einstieg in das Projekt ermöglichen. Es wurden auch Regelungen für die Quelltextformatierung definiert, um den Quelltext auch bei mehreren Entwicklern einheitlich zu strukturieren und dadurch auch lesbarer zu machen. Durch die Verwendung von Design Patterns werden die Funktionen effizient implementiert, so dass diese zukünftig erweiterbar und im Quelltext sauber strukturiert und lesbar bleiben. Unterstützt wird diese klare Strukturierung durch die Einhaltung der SOLID-Prinzipien.

Das Projekt wird in mehreren Repositories auf GitHub entwickelt. Dadurch wird auch die Entwicklung in verschiedenen Technologien getrennt, so dass der Fokus in jedem Projekt auf einer bestimmten Komponente und Technologie der Anwendung liegt.

Um die Identität eines Benutzers oder einer Benutzerin zu bestätigen, verwenden wir JSON Web Tokens (JWT). Die Kommunikation zwischen den verschiedenen Komponenten der Anwendung findet immer verschlüsselt statt. So kann die Anwendung auch in öffentlichen, potenziell unsicheren Netzwerken verwendet werden, ohne dass dabei kritische Informationen abgefangen werden können. Durch Validierung und eine rollenbasierte Rechteverwaltung soll verhindert werden, dass schädliche Informationen in das System gelangen und Benutzer:innen nur die Aktionen ausführen können, für welche diese auch zugelassen wurden.

Die Gesamtanwendung wurde in verschiedene Services aufgeteilt. Dies ermöglicht den Einsatz spezialisierter Software für die entsprechenden Aufgaben der Anwendung. Das Frontend wird mit React, Next.js und Node.js umgesetzt. Damit stehen Anwender:innen eine benutzerfreundliche Oberfläche zur Verfügung. Das Backend wird mit ASP.NET entwickelt. Darüber steht eine API zur Verfügung welche vom Frontend, über die Benutzeroberfläche, mit Informationen versorgt werden kann. Aber auch die Abfrage von Daten oder Anmeldung und Registrierung von Benutzern und Benutzerinnen findet über diese API statt. Durch den Einsatz von Swagger und Open API steht eine Dokumentation der API zur Verfügung. In einer PostgreSQL-Datenbank werden alle Daten, welche über die API übermittelt werden, gespeichert. Davon ausgenommen sind Dateien wie Bilder oder Dokumente. Diese werden in einem eigenen Speicher gespeichert. Somit wird die Datenbank von diesen Daten entlastet. Es wird mit SeaweedFS außerdem eine spezielle Software verwendet, welche auf die Speicherung von Dateien spezialisiert ist. Für den Versand von Nachrichten wird ein Mail-Server verwendet, da die Kommunikation über E-Mails direkt mit den Benutzern und Benutzerinnen stattfindet. Nachrichten in Form von Benachrichtigungen auf der Plattform oder anderen Diensten sind nicht vorgesehen.

Pakete und Komponenten

In diesem Kapitel gehen wir näher auf die verschiedenen Komponenten und Pakete näher ein. Dabei beschreiben wir auch wie die verschiedenen Komponenten und Pakete grundlegend aufgebaut sind. Diese Informationen dienen der Dokumentation grundlegender Strukturen, auf welchen die Implementierung aufsetzt und weitere Dokumentation dann konkret im Code stattfindet.

Frontend

Die Grundlage des Frontends bildet React 19.0 und Next.js 15.3.0. Damit ist die Entwicklung der Logik zur Darstellung und Ausführung des Frontends möglich. Die Verwendung dieser zwei Produkte in Kombination wird empfohlen, da bereits viele Anwendungsfälle bei der Entwicklung umgesetzt sind. Andernfalls müsste beispielsweise die Konvertierung von SCSS nach CSS oder das Server-Side-Rendering und Routing selbst umgesetzt werden. Als technische Grundlage dient dazu Node.js in der Version 22. Es handelt sich dabei um eine LTS-Version, welche noch bis Oktober 2026 unterstützt und mit Aktualisierungen versorgt wird. Der Support der direkten folgenden Version 23 wird bereits zum Mai 2025 eingestellt. Im April oder Mai 2025 soll die neue LTS-Version 24 erscheinen. Da die Implementierung der Anwendung jedoch früher beginnt wird diese Version nicht eingesetzt. Es sollte aber in Zukunft kein Problem sein von der LTS-Version 22 auf die LTS-Version 24 zu wechseln.

Die grundlegende Gestaltung findet über die Komponentenbibliothek Mantine statt. Diese bietet bereits viele Komponenten an, um Web-Anwendungen zu erstellen. Durch den konsequenten Einsatz einer solchen Bibliothek entsteht eine Web-Anwendung mit konsistentem Design, bietet aber auch die Flexibilität zukünftige Anforderungen oder neue Funktionen umzusetzen. Der modulare Aufbau dieser Bibliothek ermöglicht dabei auch die Verwendung der minimal benötigten externen Abhängigkeiten.

Die Erstellung der verschiedenen Komponenten und Seiten folgt dem Organisationsansatz nach Atomic Design. Das ermöglicht auch kleinste Änderungen, so dass diese sich aber auch in allen Komponenten und Seiten zeigen. Auch die Planung der Komponenten und Seiten in verschiedenen Tools wie Figma findet nach diesem Ansatz statt. Damit ist ein einfacher Übergang von der Planung in die Implementierung möglich.

In verschiedenen Bereichen der Anwendung werden Icons benötigt. Dafür kommt die Icon-Bibliothek Phosphor zum Einsatz, die bereits eine umfangreiche Auswahl an Symbolen bietet. Bei Bedarf können zusätzliche Icons durch eigene Implementierungen integriert werden – auch wenn dies nach Möglichkeit vermieden werden sollte, um das konsistente Design beizubehalten.

Bei der Auswahl der Bibliotheken lag der Fokus auf einer möglichst schlanken Abhängigkeitsstruktur, um spätere Aktualisierungen zentraler Komponenten ohne größere Komplikationen zu ermöglichen.

Responsive Design

Eine Anforderung an das Frontend ist das responsive Design. Dieses ermöglicht eine flexible Darstellung der Benutzeroberfläche auf verschiedenen Endgeräten. Die Oberfläche muss dazu nur einmal entwickelt werden. Durch verschiedene sogenannte Breakpoints werden Elemente der Oberfläche auf verschiedenen Auflösungen unterschiedlich angezeigt. So kann für jede Größe des Endgeräts eine passende Darstellung erreicht werden. Die Organize-It GmbH fordert die Umsetzung eines responsiven Designs für Geräte mit Bildschirmgrößen ab Tablet-Format

aufwärts, einschließlich Notebooks und Desktop-Computern. Eine Optimierung für kleinere mobile Endgeräte wie Smartphones ist nicht gefordert.

Struktur des Repositories
Im Frontend wird die folgende Verzeichnisstruktur verwendet:

- src: Der Quelltext der Anwendung.
- src/app: Die Grundstruktur der Anwendung.
- src/components: Die eigenen Komponenten, welche für den Aufbau der Anwendung benötigt werden.
- src/models: Die Entitäten im Frontend.

Backend

Das Backend wird mit ASP.NET und der Programmiersprache C# entwickelt. Ein wesentlicher Grund für die Wahl dieser Technologie ist die hohe Kompetenz des Entwicklungsteams im Umgang mit ASP.NET Core. Dadurch entfällt ein aufwendiger Einarbeitungsaufwand, und die Entwicklung kann effizient und mit hoher Codequalität erfolgen. Das Framework bietet bereits eine Struktur an, um eine API zu entwickeln und dem Frontend anzubieten. Das Frontend kommuniziert ausschließlich über diese API mit dem Backend. An das Backend sind weitere Systemkomponenten wie die Datenbank oder der Dateispeicher angebunden.

Um dem Frontend Daten zu liefern, werden zusätzlich zu den Entitäten sogenannte Data Transfer Objects verwendet. Damit werden dann auch sensible Daten, welche das Backend nicht verlassen dürfen, ausgefiltert. Die Data Transfer Objects ermöglichen aber auch ein Refactoring des Backends, ohne dass sich dabei die Strukturen aus der API ändern.

Um die Verbindung mit der Datenbank herstellen zu können, wird das Entity Framework von Microsoft eingesetzt. Als Datenbank kommt PostgreSQL in der Version 17.4 zum Einsatz.

Struktur des Repositories
Im Backend wird die folgende Verzeichnisstruktur verwendet:

- Controllers: Die Controller der API.
- Services: Die Services der API.
- Repositories: Die Repositories der API.
- Models: Die Entitäten innerhalb der API.

Datenbank

Zur Speicherung der Daten wird das Datenbankmanagementsystem PostgreSQL verwendet. Dieses System ist im Enterprise-Umfeld sehr beliebt, da dieses durch die Open-Source-Lizenzierung keine Lizenzkosten anfallen. Aber auch die verschiedensten Erweiterungen sowie die ACID-Konformität machen dieses System sehr beliebt und ein sicheres Speichern von Daten möglich.

Das Datenbankmodell ist relational aufgebaut und orientiert sich an den Anforderungen der Anwendung. Die Tabellenstruktur bildet dabei die fachlichen Entitäten sowie deren Beziehungen untereinander ab. Jede Entität besitzt eine eigene Tabelle mit einem eindeutigen Primärschlüssel, meist in Form einer UUID. Beziehungen zwischen Entitäten werden entweder durch Fremdschlüssel (bei 1:n oder n:1 Beziehungen) oder durch Zwischentabellen (bei n:m Beziehungen) modelliert. Diese Struktur erlaubt eine klare Trennung der Daten und stellt sicher,

dass Änderungen in einem Teilbereich der Anwendung keine unbeabsichtigten Seiteneffekte auf andere Bereiche haben.

Besonderes Augenmerk liegt dabei auf der Einhaltung der Normalformen, um Redundanzen zu vermeiden und Datenintegrität zu gewährleisten. Zugleich wurde bewusst darauf geachtet, dass Modell nicht unnötig zu komplex zu gestalten, um eine gute Wart- und Erweiterbarkeit sicherzustellen.

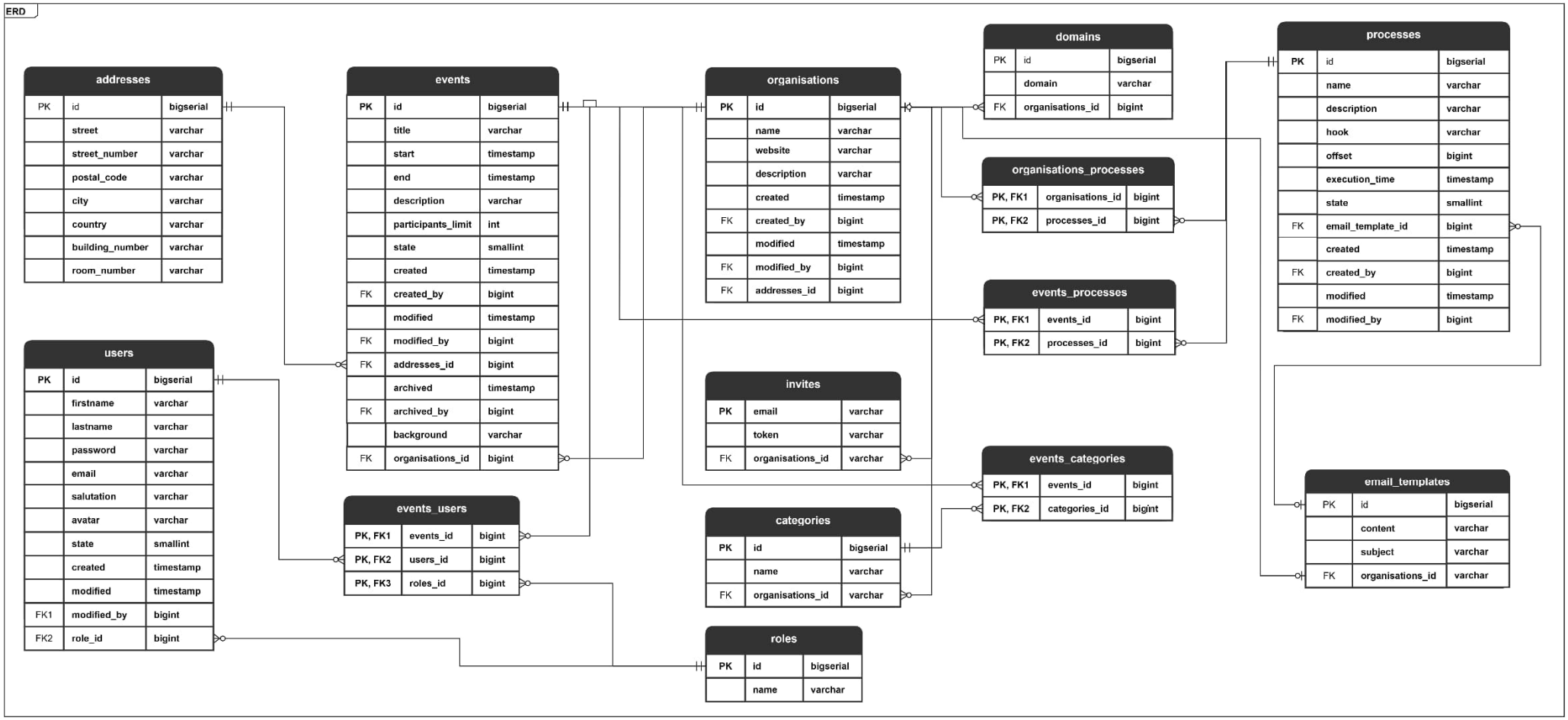

Abbildung 6: Entity-Relationship-Diagramm

Übersicht der Klassen

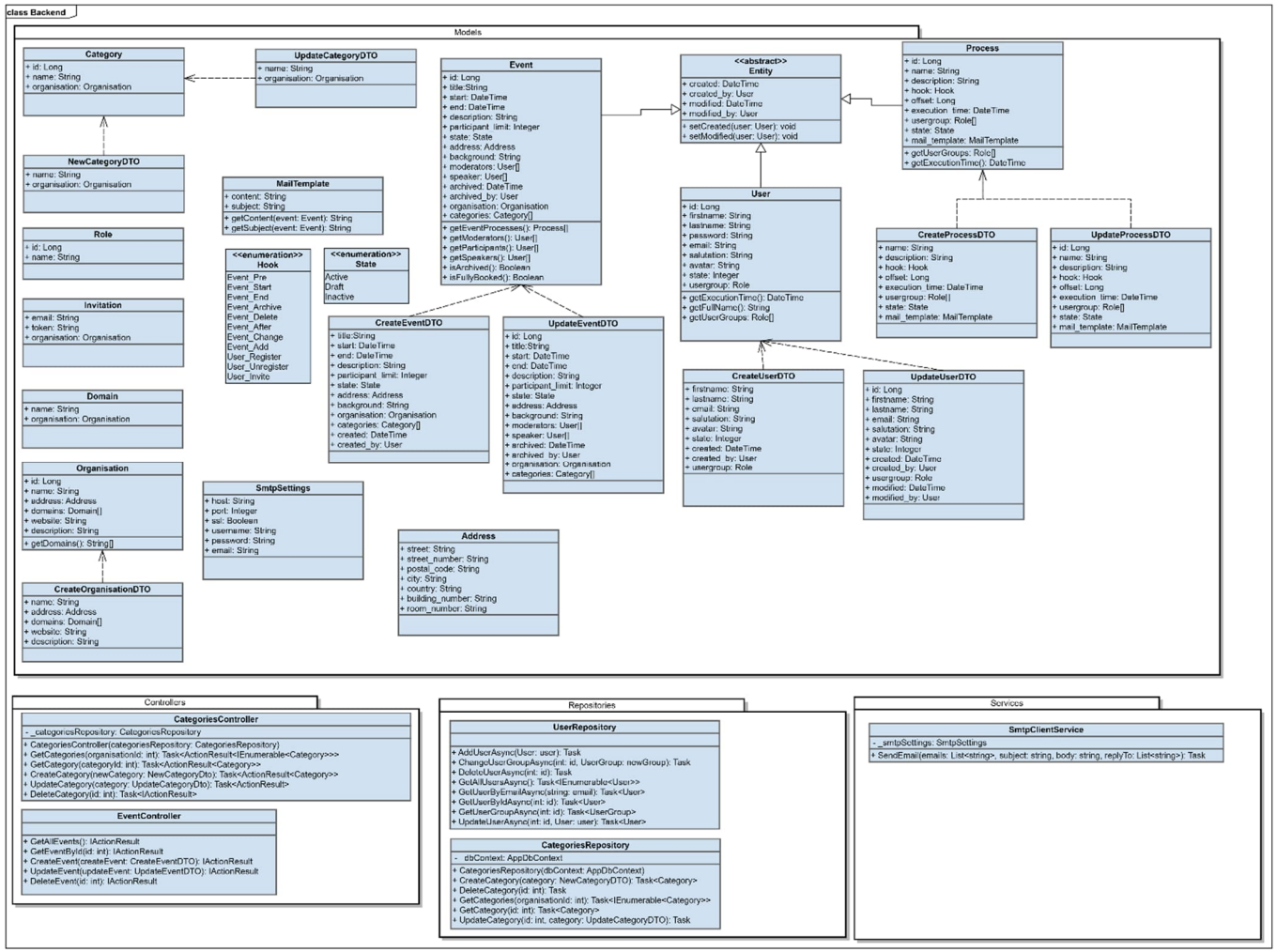

Abbildung 7: Klassendiagramm des Backends

Migrationen und Versionierung

Für die Verwaltung und Versionierung des Datenbankschemas wird EF Core Migrations (Entity Framework Core) eingesetzt. Damit kann jede Änderung am Modell nachvollzogen und bei Bedarf rückgängig gemacht werden. Jede Migration wird versioniert abgelegt und dokumentiert. Der aktuelle Zustand der Datenbank ist damit jederzeit rekonstruierbar – ein wichtiges Kriterium für Wartung und Weiterentwicklung.

Die Migrationsdateien werden im Repository gemeinsam mit dem Quellcode versioniert. Änderungen am Datenbankschema werden ausschließlich über diesen Weg durchgeführt, um Inkonsistenzen zwischen Entwicklungs-, Test- und Produktionsumgebungen zu vermeiden.

Sicherheit und Zugriff

Zugriffe auf die Datenbank erfolgen ausschließlich über das Backend. Direkte Verbindungen von außen oder vom Frontend zur Datenbank sind nicht vorgesehen. Die Datenbank ist durch ein rollenbasiertes Rechtesystem abgesichert, bei dem nur das Backend über eine dedizierte Rolle mit begrenzten Rechten auf die Datenbank zugreift. Zudem wird die Verbindung zur Datenbank verschlüsselt (TLS), um die Datenübertragung abzusichern.

Dateispeicher / File Storage

Dateien werden nicht direkt in einem Verzeichnis auf dem Server gespeichert, sondern in einem speziell dafür entwickelten Dateispeicher, basierend auf SeaweedFS. Dabei handelt es sich um ein verteiltes Dateispeichersystem, das sich durch hohe Performance, Skalierbarkeit und einfache Integration auszeichnet. SeaweedFS eignet sich besonders gut für Anwendungen, bei denen große Mengen an Dateien effizient gespeichert, verwaltet und abgerufen werden müssen.

SeaweedFS organisiert Dateien in sogenannten Volumes, die auf Volume Servern abgelegt sind. Ein zentraler Master Server verwaltet dabei die Metainformationen über die Dateien und deren Speicherorte. Die Dateien selbst werden über eine REST-API in das System geschrieben und gelesen.

In unserer Architektur erfolgt der Zugriff auf den Dateispeicher ausschließlich über das Backend. Dieses stellt eine eigene Schnittstelle zur Verfügung, über welche das Frontend Dateien hochladen, abrufen oder löschen kann. Die eigentliche Kommunikation mit SeaweedFS findet kapselt im Backend statt und wird über ein dediziertes Service-Modul gesteuert.

Jede Datei wird mit einer eindeutigen File ID versehen, die von SeaweedFS vergeben wird. Zusätzlich werden relevante Metadaten wie Dateiname, MIME-Type, Bucket-Zugehörigkeit, Erstellungsdatum und Benutzer:innen in einer zugehörigen Datenbanktabelle im Backend gespeichert. Diese Informationen erlauben eine strukturierte Verwaltung und ein schnelles Auffinden der Dateien.

Zur Strukturierung kommen auch in Verbindung mit SeaweedFS sogenannte FileBuckets zum Einsatz. Diese gruppieren Dateien logisch nach Anwendungsfällen, Benutzer:innen oder Projekten. Auf diese Weise kann das System bei wachsendem Datenvolumen effizient verwaltet und ausgebaut werden.

Um eine sichere Speicherung und Übertragung zu gewährleisten, wird die Verbindung zwischen Backend und SeaweedFS über HTTPS bzw. TLS abgesichert.

Die gesamte Zugriffskette ist durch die vorhandene Authentifizierungs- und Autorisierungslogik abgesichert. Nur berechtigte Benutzer:innen oder interne Prozesse können auf bestimmte Dateien zugreifen. Zusätzlich werden alle Dateioperationen (Upload, Löschung) mit Zeitstempel und Benutzerinformationen protokolliert, um eine lückenlose Nachvollziehbarkeit zu gewährleisten.

E-Mail-Server

Die Kommunikation in der Plattform findet ausschließlich über E-Mail-Nachrichten statt. Es gibt keine Benachrichtigungen innerhalb der Plattform selbst. Stattdessen wird jede relevante Systembenachrichtigung, wie beispielsweise Registrierungsbestätigungen, Passwortzurücksetzungen oder statusbezogene Informationen, per E-Mail an die Benutzer:innen versendet.

Für den Versand dieser E-Mails wird ein externer E-Mail-Server angebunden, der über eine SMTP-Verbindung erreichbar sein muss. Die dafür notwendigen Verbindungsdaten – wie Hostname, Port, Benutzername und Passwort – werden zentral in der Konfigurationsdatei der Anwendung hinterlegt und können je nach Umgebung angepasst werden (z. B. Entwicklungs-, Test- oder Produktionsumgebung).

Die E-Mail-Funktionalität wird ebenfalls über ein dediziertes Service-Modul im Backend abgebildet. Dieses übernimmt den Versand der Nachrichten, behandelt Fehlermeldungen beim Versand und stellt sicher, dass E-Mail-Vorlagen korrekt mit Inhalten befüllt werden.

Der Versand erfolgt standardmäßig über eine gesicherte Verbindung (TLS), um die Vertraulichkeit der übermittelten Daten sicherzustellen. Zusätzlich wird darauf geachtet, dass alle Absenderadressen klar erkennbar und vertrauenswürdig sind. Damit wird sichergestellt, dass die E-Mails korrekt zugestellt werden und nicht fälschlicherweise als Spam eingestuft werden.

Durch die ausschließliche Kommunikation per E-Mail ist die zuverlässige Funktion dieses Dienstes zentral für die Nutzerinteraktion innerhalb der Plattform. Eine kontinuierliche Überwachung sowie klare Protokollierung des E-Mail-Versands sind daher vorgesehen, um etwaige Probleme schnell erkennen und beheben zu können.

Hosting

Das Hosting der Anwendung erfolgt containerbasiert über Docker. Docker ermöglicht es, die verschiedenen Bestandteile der Anwendung – darunter Frontend, Backend, Datenbank und weitere Dienste – in jeweils eigene Container zu kapseln. Diese Container laufen isoliert voneinander, wodurch sich Änderungen oder Aktualisierungen der einzelnen Komponenten unabhängig voneinander durchführen lassen, ohne die Gesamtanwendung zu beeinträchtigen.

Die Organisation und Steuerung der einzelnen Container übernimmt ein Docker Stack. Dieser erlaubt nicht nur die Verwaltung der Container, sondern auch deren gezielte Orchestrierung. Durch den Einsatz des speziell für Container-Umgebungen entwickelten Reverse Proxys Traefik kann der Zugriff von außen flexibel geregelt werden. Öffentliche Dienste wie das Frontend sind so vom direkten Zugriff auf interne Komponenten wie die Datenbank oder das Backend entkoppelt, was die Sicherheit und Wartbarkeit der Anwendung erhöht.

Mit wachsenden Anforderungen an Performance und Skalierbarkeit lassen sich einzelne Container problemlos auf weitere Server replizieren. In Kombination mit Traefik kann die Last

dabei auch automatisch auf mehrere Server verteilt werden. Dies schafft eine flexible und zukunftssichere Infrastruktur, die auch in größeren Betriebsumgebungen bestehen kann.

Für den Prototyp wird die Anwendung vollständig containerisiert ausgeliefert. Die für den Betrieb notwendigen Docker-Images werden über bereitgestellte Konfigurationsdateien erzeugt. Damit ist eine einfache Bereitstellung in verschiedenen Umgebungen (lokal, Staging, Produktion) möglich – ohne Abhängigkeiten von spezifischen Betriebssystemen oder lokalen Installationen.

Benutzeroberfläche

Die Benutzeroberfläche der Plattform ist klar strukturiert und orientiert sich an etablierten UX-Prinzipien, um eine möglichst intuitive Bedienung zu ermöglichen – auch für technisch weniger versierte Anwender:innen. Sie basiert auf einer Seitenstruktur, die durch eine linksseitige Navigation gesteuert wird.

Navigationsbereich

Auf der linken Seite der Anwendung befindet sich eine permanente Navigationsleiste. Diese bietet Zugriff auf die wichtigsten Bereiche der Plattform. Die sichtbaren Einträge innerhalb dieser Navigation sind rollenabhängig – je nach Berechtigung einer Benutzerin oder eines Benutzers werden mehr oder weniger Tabs angezeigt. Dadurch wird die Oberfläche übersichtlich gehalten und die Nutzer:innen werden nicht mit für sie irrelevanten Funktionen konfrontiert.

Am unteren Rand der Navigationsleiste befinden sich zwei zentrale Aktionen:

- Profil anzeigen: Öffnet die Profilseite, auf der persönliche Informationen eingesehen und ggf. angepasst werden können.
- Abmelden: Meldet die aktuelle Sitzung ab und leitet zurück auf die Login-Seite.

Seitenübersicht

Die Navigationsleiste führt zu folgenden Seiten:

Übersicht

Die Übersichtsseite stellt alle verfügbaren Events dar – sowohl allgemeine Veranstaltungen als auch solche, an denen die eingeloggte Person aktiv teilnimmt. Eine integrierte Such- und Filterfunktion erleichtert das Auffinden spezifischer Events.

Alle Events

Diese Seite listet sämtliche Events der Plattform unabhängig von der eigenen Teilnahme. Auch hier stehen umfangreiche Such- und Filterfunktionen zur Verfügung, um gezielt nach bestimmten Veranstaltungen zu suchen. Die Darstellung ist besonders auf Übersichtlichkeit und einfache Navigation ausgelegt.

Verwaltung (für berechtigte Nutzer:innen)

Auf dieser Seite erhalten berechtigte Personen eine schnelle Übersicht zu aktuellen Kennzahlen. Dazu zählen Events, die in der aktuellen Woche stattfinden, die Anzahl der Besucher:innen in den letzten 30 Tagen sowie die Gesamtanzahl der sich derzeit in Planung befindenden Events.

Events verwalten (für berechtigte Nutzer:innen)

Diese Seite bietet eine tabellarische Darstellung der anstehenden Events. Dort können Veranstaltungen bearbeitet, Teilnehmer:innen eingesehen und zugehörige Prozesse angelegt

werden. Die Darstellung ist auf Effizienz in der Verwaltung ausgerichtet und unterstützt bei der operativen Durchführung.

Benutzerverwaltung (für berechtigte Nutzer:innen)
Diese Seite zeigt alle registrierten Benutzer:innen inklusive ihrer jeweiligen Rollen. So kann die Zuweisung von Berechtigungen nachvollzogen und bei Bedarf angepasst werden. Ebenso kann der Name oder die Email geändert werden. Es kann bei Bedarf eine CSV Datei hochgeladen werden, welche dann die übergebenen Benutzer anlegt.

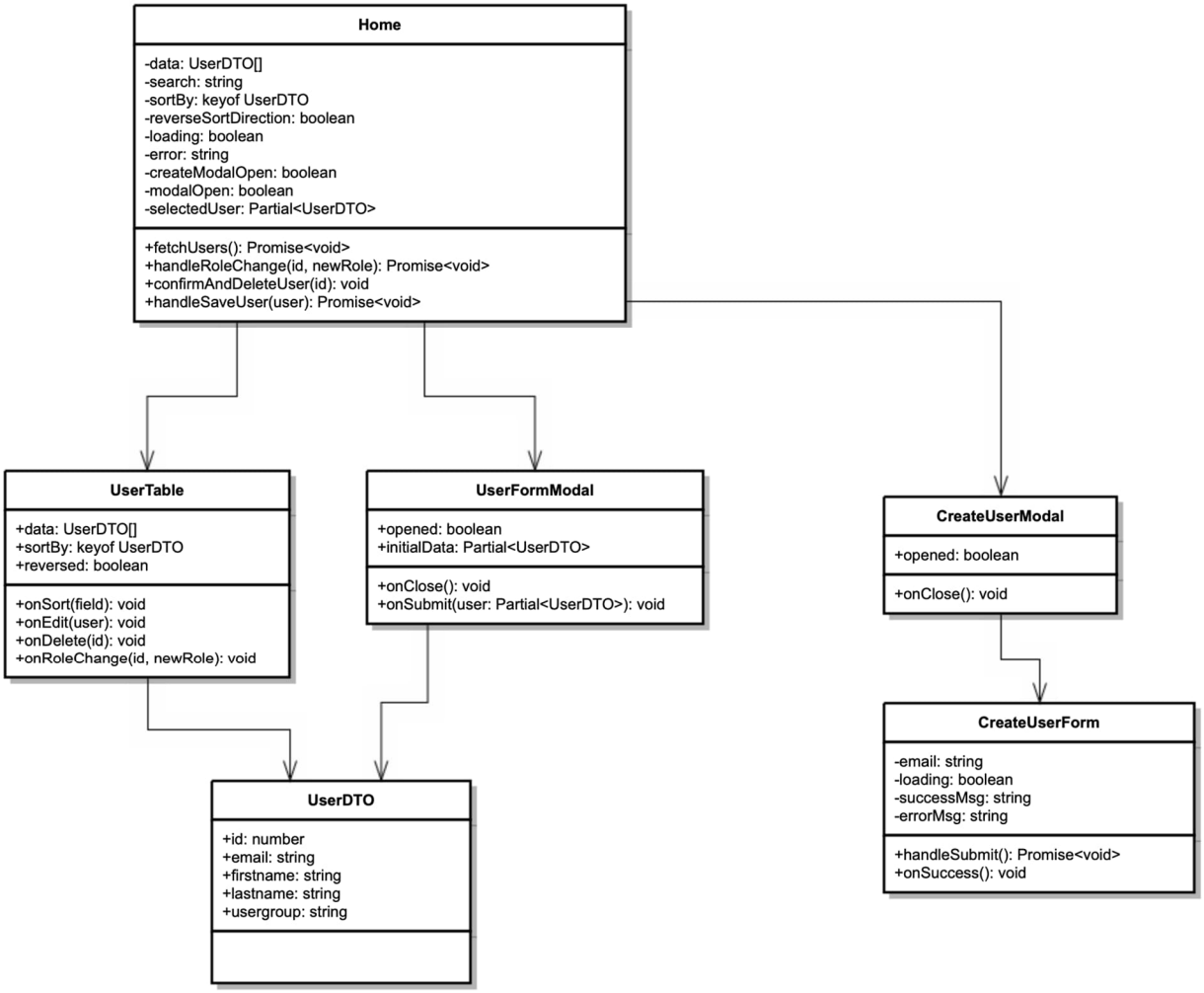

Abbildung 8: UML-Diagramm der Benutzerverwaltung

Archiv
Hier sind alle abgeschlossenen Events einsehbar. Diese Seite steht allen Nutzer:innen offen und dient als Nachschlagewerk vergangener Veranstaltungen.

Dynamischer Ablauf

Auf den folgenden Seiten befinden sich verschiedene Diagramme, welche die dynamischen Abläufe beschreiben.

Registrierung

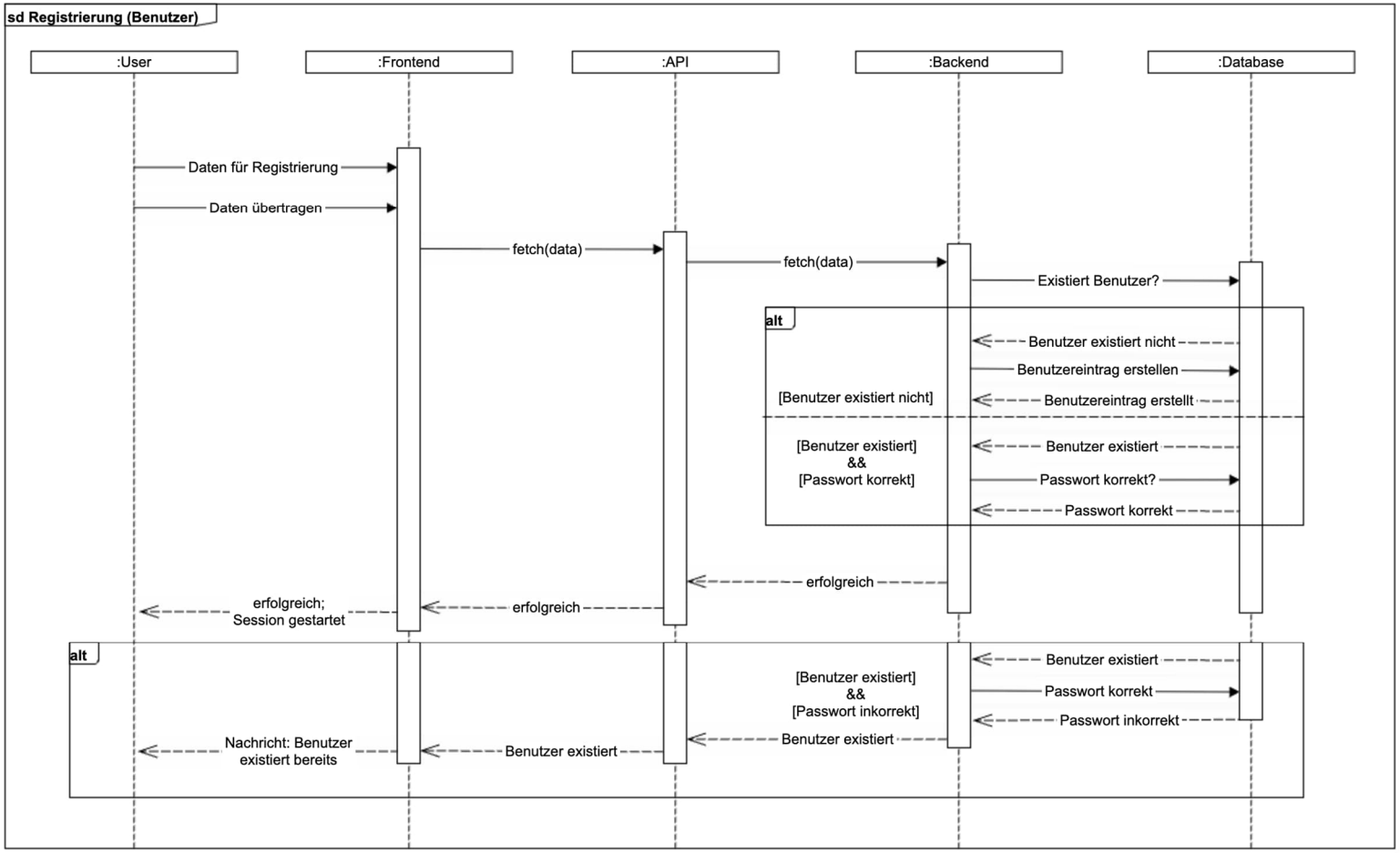

Abbildung 9: Sequenzdiagramm Registrierung

Zu beachten: Wenn der Benutzer existiert und das „neue“ Passwort auch schon richtig ist, wird der Benutzer oder die Benutzerin eingeloggt.

Anmeldung

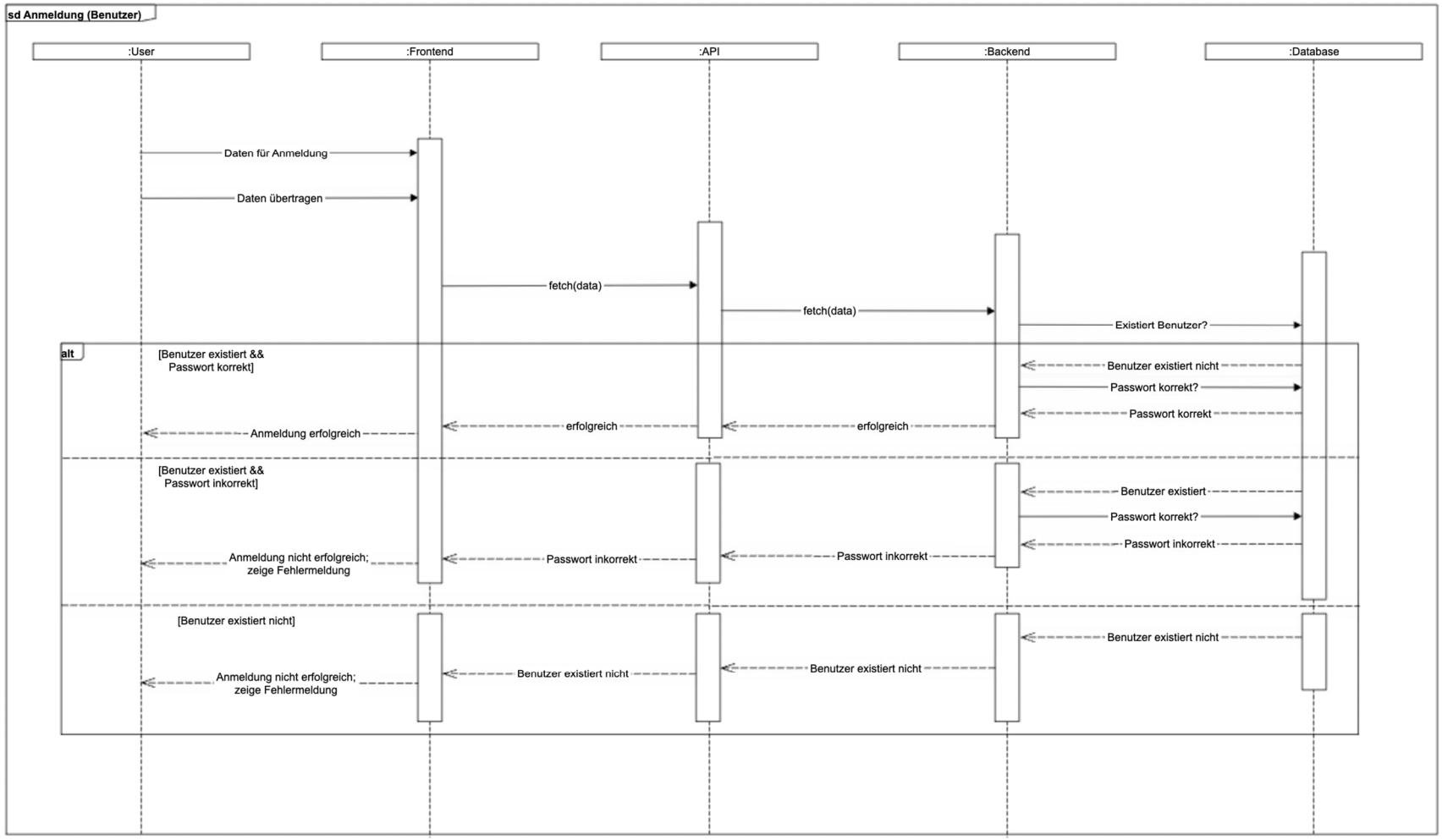

Abbildung 10: Sequenzdiagramm Anmeldung

Event löschen

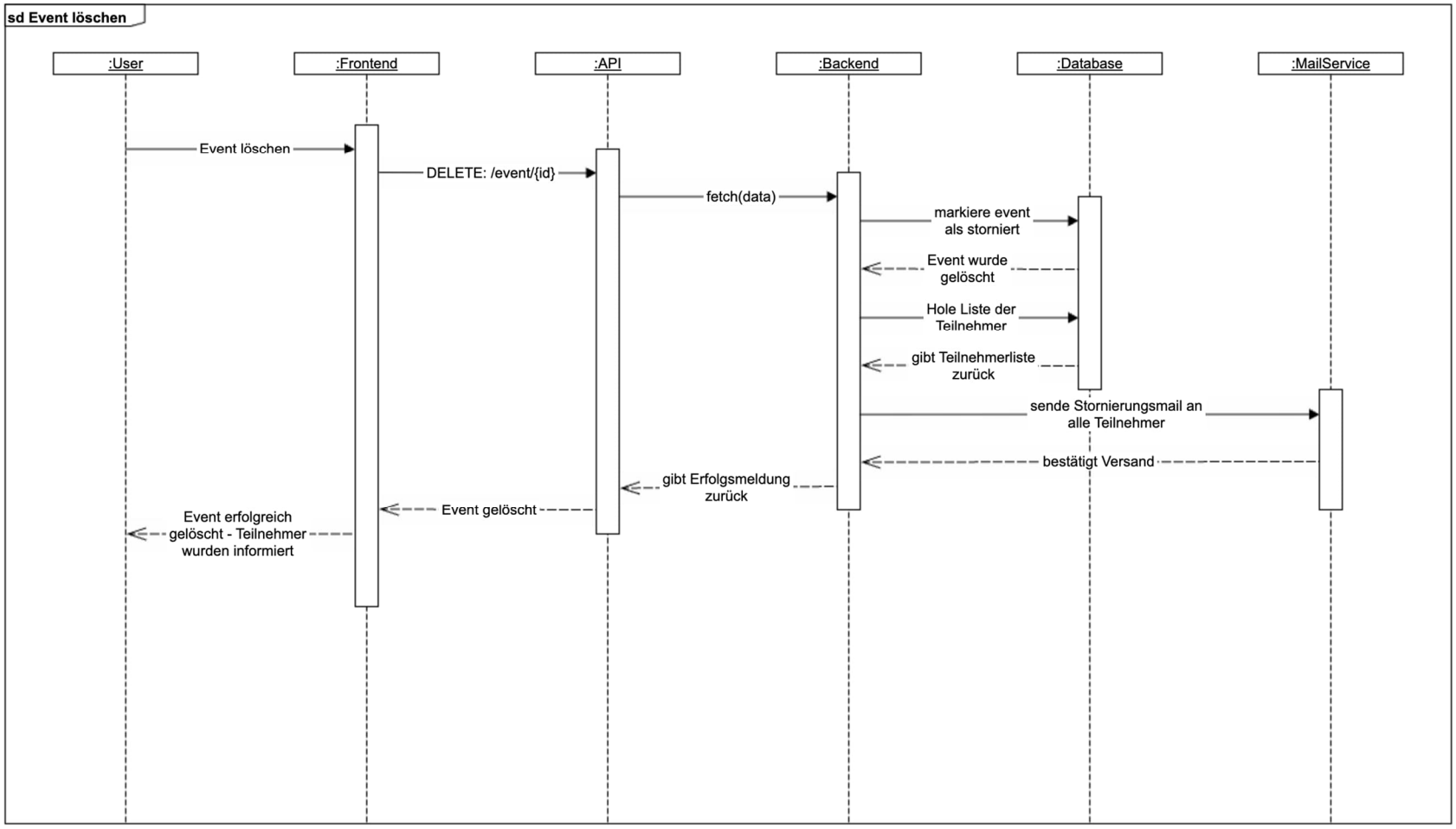

Abbildung 11: Sequenzdiagramm Event löschen

Event erstellen

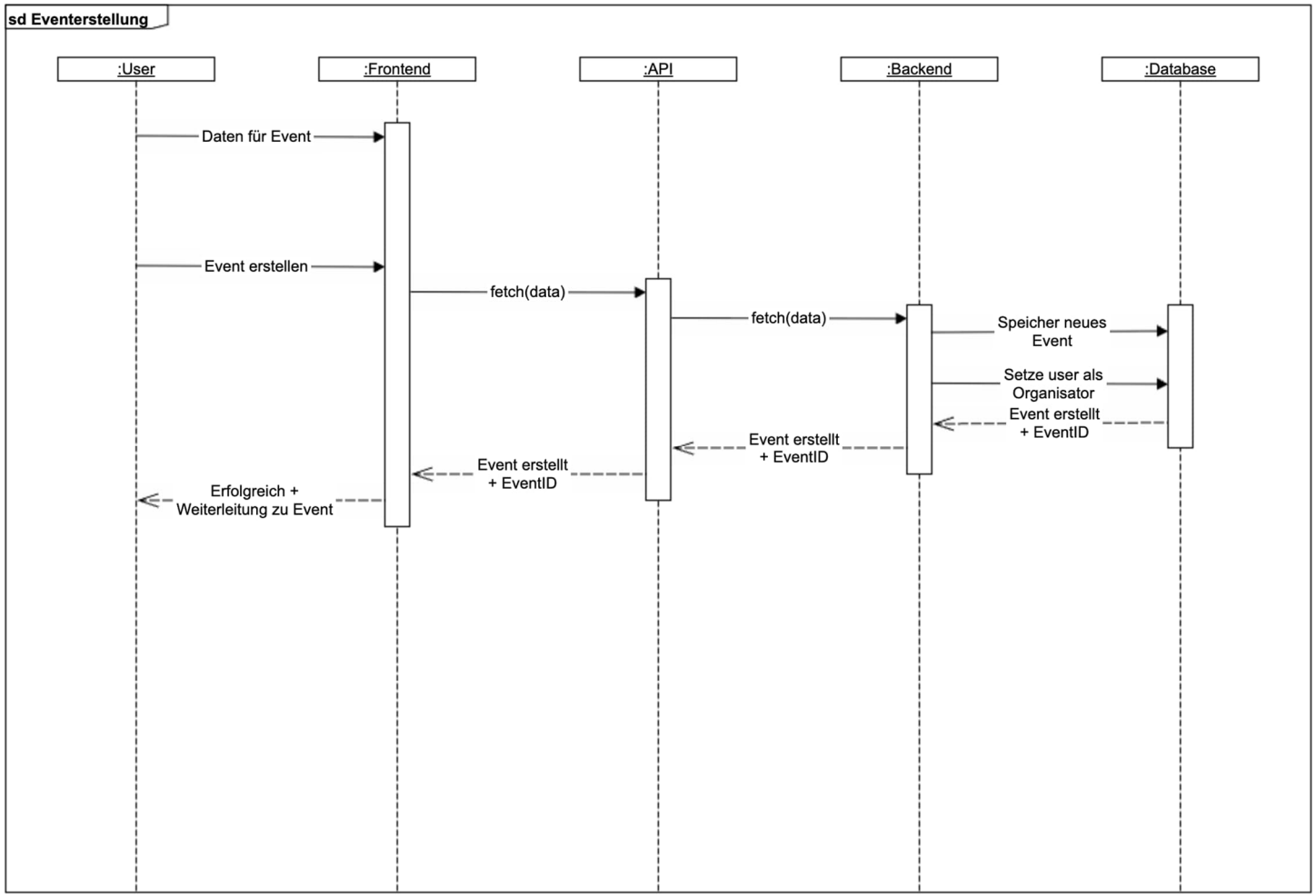

Abbildung 12: Sequenzdiagramm Eventerstellung