



Designstudie

Erstellt von:

- **Robin Schnaiter**

Verifiziert von:

- **Fabian Hamacher**

Version	Erstellung	Beschreibung
0.1	06.04.2025	Erstellung Dokument
0.2	23.03.2025	Erstellung Vorlage



Inhalt

Allgemeines	3
Produktübersicht	4
Grundsätzliche Struktur- und Entwurfsentscheidungen für die Anwendung	5
Grundsätzliche Struktur- und Entwurfsentscheidungen der einzelnen Pakete / Komponenten	6
Frontend (Next.js / React / Shadcn)	6
Backend (ASP.NET Core)	6
Schnittstellen.....	6



Allgemeines

Kurzbeschreibung des Projekts: Entwicklung eines Online-Eventmanagement-Systems im Auftrag der Organizelt GmbH. Die Software ermöglicht es Organizelt, interne physische Veranstaltungen für ihre Unternehmenskunden effizient zu planen, zu konfigurieren und zu verwalten. Der Fokus liegt auf Benutzerfreundlichkeit sowie der Automatisierung von Prozessen wie Teilnehmerregistrierung und Eventkommunikation. Nach Abschluss des Projekts wird die Software von Organizelt genutzt, um Events für verschiedene Unternehmen zu hosten.

- **Tech-Stack:**

Frontend: Next.js + React

Backend: ASP.NET Core

Datenbank: PostgreSQL

APIs: REST API mit verschlüsselter Kommunikation

- **Motivation für die Wahl der Technologien:**

- **Next.js & React:**

Die Entscheidung für React fiel, weil im Team bereits bei einigen Mitgliedern ein Grundverständnis vorhanden war. Aufgrund der vergleichsweise flachen Lernkurve im Vergleich zu komplexeren Frameworks wie Angular, ist React die bessere Wahl, um schnelle Fortschritte zu erzielen und eine hohe Team-Effizienz zu gewährleisten.

Durch den Einsatz von Next.js wird zusätzlich die Performance des Frontends erhöht, und die Entwicklung von gut strukturierten, skalierbaren Anwendungen wird vereinfacht.

- **ASP.NET Core:**

Die Backend-Entwicklung wurde auf ASP.NET Core ausgerichtet, da die Kernkompetenzen des Teams im Bereich C# und .NET liegen. Dadurch kann das Team auf bestehendes Wissen zurückgreifen, was die Produktivität steigert und die Wartbarkeit des Codes erleichtert. Außerdem bietet ASP.NET Core eine hohe Performance, integrierte Sicherheitsfeatures und eine ausgezeichnete Unterstützung für eine modulare, skalierbare Architektur.

- **REST API:**

Als Kommunikationsschnittstelle zwischen Frontend und Backend wurde REST gewählt, da es sich um einen weit verbreiteten Standard handelt, der einfach zu implementieren und gut dokumentierbar ist (z. B. mit Swagger). REST bietet zudem eine klare Trennung der Verantwortlichkeiten und erleichtert die Integration weiterer Clients (z.B. mobile Apps) in der Zukunft.



Produktübersicht

- **Beschreibung der äußerlichen Funktionsmerkmale:** Benutzerfreundliches Interface für Organisatoren und Teilnehmer, optimiert für Tablets, Desktop-first-Ansatz, keine Smartphone-Optimierung erforderlich.
- **Benutzerführung:**
 - Dashboard für Organisatoren: Eventverwaltung, Teilnehmerlisten, Kategorienverwaltung, Vorlagenmanagement.
 - Teilnehmerbereich: Eventübersicht, Filter- und Suchfunktion, Anmeldung/Abmeldung.
 - Event-Detailseiten: Informationen zum Event, Anmeldung, Upload von Materialien (optional).
 - Administrationsbereich: Organisationen verwalten, Benutzerrechte definieren.
- **Interaktionen:**
 - Event erstellen, konfigurieren und verwalten.
 - Teilnehmerregistrierung und Verwaltung.
 - Einladung per spezifischem Link.
 - Upload/Download von Event-Dokumenten.
 - Automatisierte E-Mail-Benachrichtigungen.
 - Nutzung von Prozessvorlagen.
 - Filter- und Suchfunktionen.
- **Zielgruppen:**
 - Primär: Organisatoren von Events innerhalb von Unternehmen.
 - Sekundär: Teilnehmer, Speaker, Moderatoren, externe Gäste.
- **Mockups/Screenshots:**

Turbo Events

Verwaltung

Events

Mitglieder

Archiv

Turbo Events > Events

John Doe

Alle Events der Organisation




Name	Start	Ende	Ort	Beschreibung	Aktionen		
Event 1	6.4.2025 20:52	6.4.2025 22:52	Location 1	Description for Event 1	Prozesse	Teilnehmer	
Event 2	7.4.2025 20:52	7.4.2025 23:52	Location 2	Description for Event 2	Prozesse	Teilnehmer	
Event 3	8.4.2025 20:52	9.4.2025 0:52	Location 3	Description for Event 3	Prozesse	Teilnehmer	

Abbildung 1 Screenshot Eventübersicht

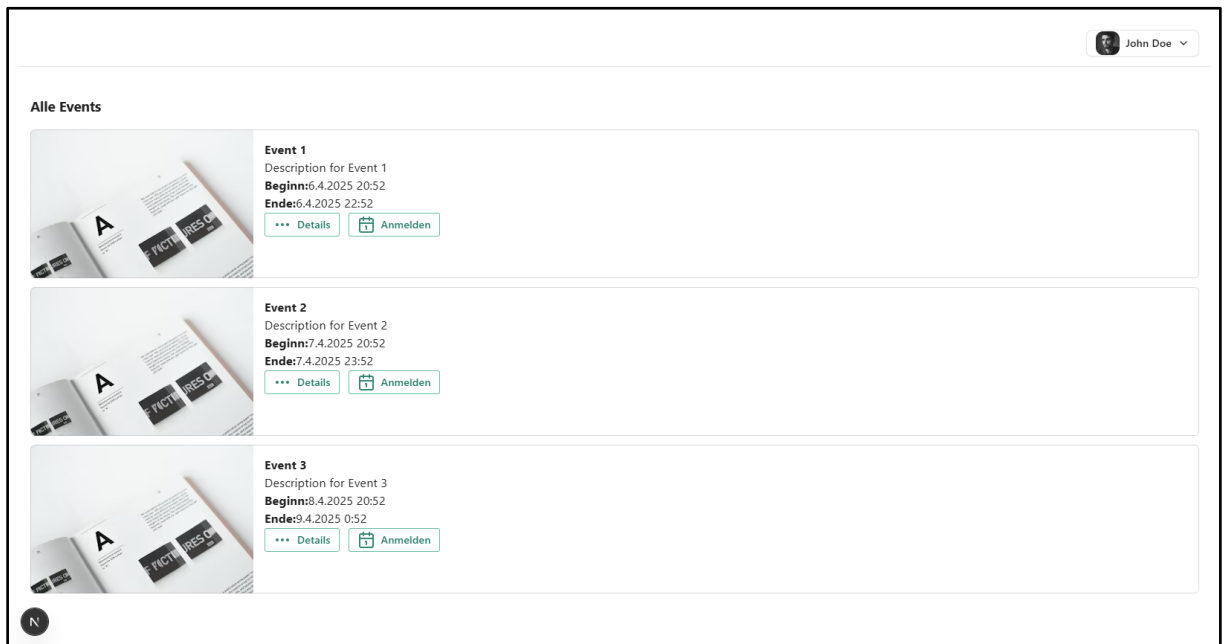


Abbildung 2 Screenshot Events

Für weiterführende Informationen zum aktuellen Stand des Prototyps wird auf das Benutzerhandbuch verwiesen.

Grundsätzliche Struktur- und Entwurfsentscheidungen für die Anwendung

- **Architekturüberblick:**
 - Client-Server-Modell mit REST API.
 - Aufgabenverteilung: Frontend übernimmt UI und Client-seitige Logik; Backend verarbeitet Geschäftslogik, Datenhaltung und API-Bereitstellung.
 - Verschlüsselte Kommunikation (HTTPS).
- **API-Kommunikation:**
 - REST API
 - Swagger/OpenAPI-Dokumentation sinnvoll.
- **Designprinzipien:**
 - Clean Code Prinzipien
 - SOLID-Prinzipien im Backend
- **Projektstruktur:**
 - Multi-Repository, da Frontend und Backend entkoppelt.
 - Ordnerstruktur Frontend: /src, /app, /(auth), /(site).
 - Ordnerstruktur Backend: /Controllers, /Services, /Repositories, /Models.
- **Security-Überlegungen:**
 - Authentifizierung & Autorisierung: JWT Token.
 - Schutz vor Angriffen: HTTPS, Validierung, Rollenbasierte Rechteverwaltung.



Grundsätzliche Struktur- und Entwurfsentscheidungen der einzelnen Pakete / Komponenten

Frontend (Next.js / React / Shadcn)

- **Seitenstruktur:**
 - Home / Login
 - Dashboard
 - Event-Detailseite
 - Event-Erstellung / -Bearbeitung
 - Adminbereich (Organisationen & Benutzerverwaltung)
- **Komponentenstruktur:**
 - Atomic Design Ansatz: Atoms (Buttons, Inputs), Molecules (Formulare), Organisms (Event-Karten), Templates (Eventseiten), Pages.
 - Verwendung von Shadcn für UI-Komponenten.
- **Styling-Ansatz:**
 - Tailwind CSS.
- **Fehlerbehandlung und Logging**
 - Logging wird an das Backend übergeben
- **Routing:**
 - Dateibasiertes Routing von Next.js.
- **API-Aufrufe:**
 - Nutzung von Fetch zum Aufruf der REST API.

Backend (ASP.NET Core)

- **Architektur:**
 - Layered Architecture: Controller / Service / Repository Layer.
 - Dependency Injection: Ja, ASP.NET Core unterstützt das nativ.
- **Datenbankzugriff:**
 - Entity Framework Core
- **Fehlerbehandlung und Logging:**
 - ASP.NET Logging Middleware nutzen.
 - Exception Handling Middleware einbauen.
 - Logging wird in Datenbank abgespeichert für spätere Fehleranalyse
- **DTOs & Mapping:**
 - Einsatz von DTOs zur Trennung von API und modelle aus der DB.
 - Automapper für Mapping verwenden.
- **API-Strukturierung:**
 - RESTful Endpoints.
 - API-Dokumentation über Swagger/OpenAPI

Schnittstellen

- **Definition der APIs:**



- REST API mit klar definierten Endpunkten für Events, Organisationen, Benutzerverwaltung, Prozesse.
- Nutzung von Swagger zur Dokumentation