
Projekt: **Event-Tool**

# **Prototyp-Designbeschreibung –** v1.0

Abgabedatum: 06.04.2025

Projektgruppe: BitWorks
Ansprechpartner: Tobias Schuhmacher – *Projektleiter*

# Inhaltsverzeichnis



# Allgemeines

Diese Dokumentation dient Software-Entwicklern, welche sich mit der Demo/Software von BitWorks beschäftigen möchten, um den Quellcode und die verwendete Infrastruktur zu verstehen und eventuell selbst Änderungen vornehmen zu können. Dazu werden die äußerlichen Funktionsmerkmale des Produkts beschrieben. Darauffolgend werden die interne Struktur und die dabei getroffenen Entwurfsentscheidungen erläutert und begründet.

# Äußerliche Funktionsmerkmale

Unter den Äußeren Funktionsmerkmale versteht man die verschiedenen Features und Interaktionsmöglichkeiten innerhalb der Software. Sie beschreiben, was ein potenzieller Nutzer des Produkts sieht und ausführen kann.

## Benutzeroberfläche

Öffnet ein Nutzer die Software, wird er auf eine Anmeldeseite weitergeleitet. Dort hat er die Möglichkeit, sich mithilfe eines bestehenden Accounts anzumelden (E-Mail und Passwort), oder über einen weiteren Button einen neuen Account zu erstellen. Wählt der Nutzer den Button „neuen Account erstellen“ wird er auf eine neue Seite geleitet, wo er Informationen (Vor- und Nachname, E-Mail, Geburtsdatum, Passwort) für seinen neuen Account eingeben kann. Betätigt der Nutzer nach Eingabe der relevanten Informationen über den Button „Registrieren“ (oder meldete sich im anderen Fall über einen bereits erstellten Account an), wird er auf die Hauptseite der Software geleitet.

Auf der Hauptseite erhält der Nutzer eine Übersicht über die verschiedenen Events, welche von seiner Organisation geplant sind. Außerdem gibt es eine Filterfunktion mithilfe von 2 Checkboxen für die Events, die es dem Nutzer ermöglichen, die Events anzuzeigen, bei denen er angemeldet ist, als auch welche, die bereits vorbei sind (und die der Nutzer besucht hatte oder auch nicht). Ist der Nutzer ein Organisator, befindet sich auf der Hauptseite ein Button, um zur Übersicht über die Organisation zu gelangen, wo er die Liste der vorhandenen Organisatoren sieht und auch neue Organisatoren einladen kann. Auf der Hauptseite befindet sich außerdem ein Button, mit welchem ein neues Event erstellt werden kann.

Drückt der Nutzer auf eines der aufgelisteten Events, wird er auf eine neue Seite mit den relevanten Informationen über das Event weitergeleitet. Dazu gehören beispielsweise das Datum und der Ort des Events. Dort gibt es auch den Button, um sich für das Event an- oder, wenn bereits angemeldet, abgemeldet werden kann. Ein Organisator sieht hier zusätzlich die Prozessschritte, die dem Event zugeordnet wurden. Er hat zusätzlich die Option, die Informationen über das Projekt über einen Button „Bearbeiten“ zu ändern. In dieser Übersicht kann er auch weitere Prozessschritte zuweisen.

Erstellt ein Organisator ein neues Event oder bearbeitet ein existierendes, wird er jeweils auf dieselbe Seite weitergeleitet. Dort kann er die Informationen des Events bearbeiten. Möchte er die Seite verlassen, kann er dies durch den Button „Änderungen Speichern“, welcher die Änderungen speichert, oder durch den Button „Änderungen Verwerfen“ (nur bei einem bereits existierenden Event) tun.

# Struktur- und Entwurfsentscheidungen

## Allgemeines zur Struktur

Im weiteren Verlauf werden die Struktur- und Entwurfsentscheidungen der gesamten Software im Überblick behandelt. Es werden die verschiedenen Klassen im Projekt und deren Verknüpfungen/Schnittstellen genannt, sowie ihre Wahl genauer erläutert. Dazu werden die verwendeten Frameworks und Programmiersprachen genannt.

### Blazor/Razor Pages

Der Prototyp wurde mithilfe von Blazor und Razer Pages entwickelt. Es erleichterte die Erstellung der verschiedenen HTML-Seiten sowie die Verknüpfung dieser untereinander, als auch mit C# Logik. Das Tech-Stack bleibt damit reduziert und erleichtert die Integration des Frontend mit dem Backend.

### C#

Bei der verwendeten Programmiersprache handelt es sich um C# mit .NET 8.0. Diese Entscheidung wurde aufgrund mehrerer Faktoren getroffen:
Wie die im Voraus durchgeführte Recherche ergeben hat, verwenden einige Konkurrenzprodukte ebenfalls C#. Dazu kommt, dass Microsoft eine gute und ausführliche Dokumentation für diese Sprache zur Verfügung stellt. Im Team sind außerdem bereits Vorkenntnisse vorhanden, was die Verwendung vereinfacht.

### Bootstrap

Das Styling wurde größtenteils mit Bootstrap erledigt. Der Grund dafür ist simpel, da es gut mit Blazor funktioniert und bereits mit der Verwendung davon zur Verfügung steht.

## Gesamtarchitektur der Komponenten

Der Prototyp besteht aus mehreren Klassen/Komponenten, welche miteinander verknüpft sind. Dieser Abschnitt beschreibt ihre Verknüpfung untereinander und begründet sie.
Es gibt folgende Klassen: Application, Event, Organization, Person, Member, Organizer, EventOwner, Right, Member, Organizer und EventOwner.

### Klassen allgemein

Die Klasse Application steht im Zentrum der Architektur. Sie besitzt Schnittstellen zu den Klassen Event, Organization und Person und speichert Instanzen dieser Klassen. Neben ihr steht die Klasse Organization, welche Instanzen der Klassen Event und Person besitzt. Des Weiteren sind die Klassen Event und Person mit einer Beziehung zueinander verknüpft: Event speichert Instanzen der Klasse Person. Die Klasse Person besitzt zudem Teil-Instanzen aus den Klassen Organization und Event (jeweils die ID dieser Objekte). Diese Verknüpfung sorgt dafür, dass die Entwickler es leichter haben, aus jeder Instanz weitere Instanzen ziehen zu können. Es macht es einfacher, nach dem Wechsel von einer Razor Page zu einer anderen, mithilfe der einzigen weitergegebenen Instanz Zugriff auf alle verknüpften Instanzen zu erhalten.

### Rechtesystem

Die Rechte der einzelnen Nutzer wurden auf drei Rollen verteilt. Diese Rollen erben von der abstrakten Klasse rights, welche alle vorhandenen Berechtigungen umfasst. In den einzelnen Rollen wird dann definiert, welche Rolle gewisse Berechtigungen erhält und welche nicht. Dieses System basiert auf dem sogenannten RBAC (Role-Based Access Control). Es handelt sich hierbei um das Strategy Pattern.

# Struktur- und Entwurfsentscheidungen einzelner Pakete/Komponenten

In diesem Abschnitt werden die Klassen/Komponenten selbst genauer untersucht und deren Aufbau begründet. Dabei wird auf ihre Funktion, eingegangen, sowie verwendete Design-Patterns erklärt und wichtige Variablen genannt.

## Application.cs - Singleton

Die Klasse Application ist als Singleton konzipiert und fungiert damit im aktuellen Prototyp als zentraler Speicher für alle Organisationen, Personen und Events. Sie stellt keine eigene Logik bereit, sondern dient ausschließlich zur temporären Datenhaltung während der Entwicklungsphase. In der finalen Applikation wird anstelle dieser Klasse eine persistente Datenbankanbindung verwendet.

## Organization

Die Klasse Organization verwaltet eine Vielzahl an Attributen und enthält insbesondere zwei voneinander getrennte Listen: eine für Personen mit der Rolle "Organizer", die erweiterte Rechte innerhalb der Organisation besitzen, und eine weitere für Member. Die explizite Trennung der Rollen über separate Listen ermöglicht eine klare Rechteverwaltung innerhalb der Organisation. Auch gibt es eine Liste für Events, welche in der Organisation veranstaltet werden. Sie stellt im Ganzen eine Organisation im Prototyp dar.

## Person

Auch die Klasse Person ist ähnlich strukturiert, enthält jedoch lediglich eine Referenz-ID auf ihre zugehörige Organisation, anstatt eine Liste von Organisationen zu verwalten. Zusätzlich führt sie eine Liste von Events, an denen die Person beteiligt ist. Diese Events sind ebenfalls nur über ihre ID referenziert. Der Grund hierfür war die Verhinderung eines „Loops“ innerhalb der Variablen (Beispiel: Person speichert Event -> Event speichert Person. Solch eine Konstellation hatte bei dem Code des Prototyps zu Problemen geführt). Aus Sicherheitsgründen wird das Passwort nicht im Klartext gespeichert, sondern ausschließlich als Hash abgelegt. Die Authentifizierung erfolgt über die Kombination aus E-Mail-Adresse und Passwort. Im Prototyp wird hiermit ein Benutzer dargestellt. Er kann verschiedene Rollen zugewiesen haben.

## Event

Die Klasse Event verwaltet sowohl eine Liste von Member als auch eine Liste von EventOwner. Sobald ein Event erstellt wird, wird die erstellende Organisator automatisch als EventOwner eingetragen. Dieses Vorgehen stellt sicher, dass jedes Event stets eine verantwortliche Person besitzt, die die administrative Kontrolle über das Event innehat. Mit dieser Klasse werden die von Organisationen erstellten Events dargestellt.

## Rechte- und Rollenmodell

Das Rechte- und Rollenmodell der Applikation umfasst drei klar definierte Rollen: Organizer, EventOwner und Member. Beim Anmeldeprozess wird überprüft, welcher Organisation eine Person zugeordnet ist und welche Rolle sie innerhalb dieser Organisation einnimmt. Diese Information wird anschließend im Session Storage abgelegt, um während der Sitzung schnell und effizient rollenbasierte Berechtigungen prüfen und anwenden zu können. Die Rollen ermöglichen die klare Strukturierung der Hierarchie unter den verschiedenen Personen in einer Organisation und bestimmt, wer welche Aktionen und Funktionen innerhalb im Prototyp verwenden darf.