Event-Planer

Felix Hoffmann, Baran Bickici, Sami Gökpinar, Ergün Bickici

20. Mai 2025

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einführung

Im Rahmen des Softwareprojekts im 4. Semester des Studienganges Softwaretechnik und Medieninformatik wird über das ganze Semester ein Projekt mit einem Partnerunternehmen, die als Kunden agieren durchgeführt. Dieses Projekt wird mit dem Unternehmen pep.digital die sich ein Produkt wünschen mit dem Titel "Event-Planer". Dieses Produkt soll eine Web-Applikation werden, in denen Mitarbeiter des Unternehmen pep.digital Events erstellen und das Event planen können z.B. mit Umfragen, Datum des Events und Teilnehmer. Dazu hat man auch die Möglichkeit Wünsche zu äußern aus denen man Events erstellen kann. Als erstes wird sich erst auf Schulungsevents konzentriert und in der Zukunft hat man die Möglichkeit auch andere Arten von Events zu erstellen.

2 Technologien

2.1 Full-Stack Framework

Für die Umsetzung des Projekts wurde Next.js als Full-Stack-Framework gewählt. Next.js basiert auf React und bietet eine vollständige Lösung für die Entwicklung von Webanwendungen, indem es sowohl Frontend- als auch Backend-Funktionalitäten integriert. Mit Next.js können Entwickler sowohl serverseitiges Rendering (SSR) als auch statische Seitengenerierung (SSG) nutzen. Zudem bietet es eine einfache Möglichkeit, API-Routen zu erstellen, was es zu einer hervorragenden Wahl für Full-Stack-Entwicklungen macht. Die Verwendung von TypeScript sorgt für eine typsichere und fehlerarme Entwicklung, sowohl im Frontend als auch im Backend.

2.1.1 Frontend

Im Frontend nutzt Next.js React. React ist eine quelloffene JavaScript-Bibliothek, die das Erstellen der Benutzeroberfläche schnell und dynamisch macht. Anhand von React können Web- und Mobile-Anwendungen mit derselben Codebasis erstellt werden, ohne jegliche Formatierung. Die Codierung erfolgt in TypeScript, was durch statische Typisierung und bessere Fehlererkennung das Entwickeln sicherer und effizienter macht.

2.1.2 Backend

Das Backend in Next.js wird durch das integrierte API-Routing realisiert, das auf Node.js basiert. Durch diese Integration können Entwickler API-Routen direkt innerhalb der Next.js-Anwendung erstellen, ohne einen separaten Server benötigen zu müssen. Diese Routen ermöglichen es, serverseitige Logik auszuführen, wie etwa das Abrufen von Daten aus einer Datenbank oder das Bearbeiten von Anfragen. Da Next.js auf Node.js aufbaut, können alle leistungsfähigen Funktionen von Node.js genutzt werden, während TypeScript die Entwicklung durch statische Typisierung und Fehlererkennung verbessert. Dies sorgt für eine konsistente und sichere Entwicklung sowohl im Frontend als auch im Backend innerhalb derselben Codebasis.

2.2 Datenbank

Für die Umsetzung der Anwendung wurde eine relationale Datenbank auf Basis von PostgreSQL gewählt. Diese Entscheidung basiert auf der strukturellen Beschaffenheit der Daten sowie auf Vorkenntnissen im Umgang mit relationalen Datenbanksystemen. Die Anwendung verarbeitet klar strukturierte Daten wie Nutzerinformationen, Veranstaltungsdetails, Wunschvorschläge und Umfrageergebnisse. Viele dieser Entitäten weisen ähnliche Attribute auf oder stehen in logisch definierten Beziehungen zueinander. Relationale Datenbanken eignen sich in solchen Szenarien besonders gut, da sie eine eindeutige Abbildung von Beziehungen ermöglichen. Zudem liegen bereits umfangreiche Erfahrungen im Umgang mit relationalen Datenbanken vor, unter anderem durch vergangene Vorlesungen und Projekte mit SQL. Der Einsatz einer vertrauten Technologie ermöglicht eine effizientere Umsetzung des Datenmodells, ohne zusätzlichen Einarbeitungsaufwand in alternative Datenbankkonzepte. PostgreSQL wurde als Datenbankmanagementsystem gewählt, da es als stabile und gut dokumentierte Lösung gilt. Es deckt sowohl grundlegende Anforderungen wie CRUD-Operationen als auch komplexere Anforderungen hinsichtlich Datenintegrität ab. In Kombination mit der Backend-Plattform Supabase wird die Datenbank um Zugriffskontrollen ergänzt, ohne dass dafür viel eigene Backend-Logik nötig ist.

2.3 Authentifizierung

Supabase wird für die Authentifizierung genutzt, da es eine einfache und sichere Möglichkeit bietet, Nutzer in eine Anwendung zu integrieren. Es unterstützt Single Sign-On (SSO) und externe Identitätsanbieter wie Google und Microsoft Azure, sodass sich Benutzer bequem mit ihren bestehenden Konten anmelden können. Durch die Integration von Microsoft Azure können sich insbesondere Mitarbeiter direkt mit ihren Microsoft-Accounts authentifizieren. Dies erleichtert den Zugriff auf die Anwendung, da keine separaten Anmeldeinformationen erstellt werden müssen, und sorgt gleichzeitig für höhere Sicherheit und bessere Verwaltungsmöglichkeiten durch zentrale Benutzerkontrollen in Azure.

2.4 Bereitstellung

Zur Bereitstellung wird in diesem Projekt Docker verwendet, da so sicher gegangen werden kann, dass die Anwendung in einer konsistenten Umgebung ausgeführt wird, unabhängig von den zugrunde liegenden Betriebssystemen oder Hardwarekonfigurationen.

3 Requirements Specification

3.1 Zielgruppe

Mit dem Endprodukt des Projekts, wird den Mitarbeitenden des Unternehmen pep.digital eine Web-Applikation für die Erstellung und Planung von Events zur Verfügung gestellt. Da diese Web-Applikation firmenspezifisch und nicht für den offenen Markt entwickelt wird, ist die Zielgruppe dieser App sehr reduziert. Dies bedeutet im Umkehrschluss, dass ähnliche Applikationen und Konkurrenten bei der Entwicklung dieser App keine Rolle spielen. Durch die Analyse bereits bestehender Anwendungen, wurden gelungene Eigenschaften und Makel aufgelistet. Dadurch ist klar, worauf es bei der Entwicklung geachtet werden soll

3.2 Kundenbefragung

Eine Kundenbefragung bei der Softwareentwicklung ist wichtig, um die Bedürfnisse und Erwartungen der Zielgruppe zu verstehen. Durch direktes Feedback von potenziellen Nutzern können Entwickler ihre Produkte besser anpassen und optimieren. Im Gespräch mit dem Kunden ergaben sich beispielsweise Informationen über den Prozess der Erstellung der Events und Wünsche.

3.3 Feature-Map

Aus der Kundenbefragung ergab sich diese Feature-Map:

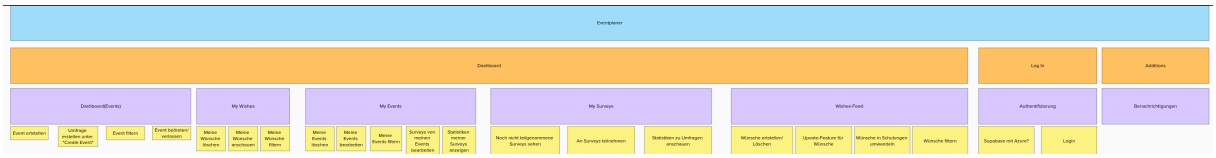

Abbildung 1: Feature-Map

3.4 User Stories

3.4.1 Anmeldung

- **Als Benutzer**, möchte ich mich mit einem bestehenden Account (z.B. Microsoft) anmelden können, **damit ich** mich nicht neu registrieren muss und die Applikation direkt mit meinem persönlichen Profil nutzen kann.
- **Als Benutzer** möchte ich mich mit meiner eigenen E-Mail-Adresse und einem selbstgewählten Passwort registrieren können, **damit ich** ein individuelles Benutzerkonto erstellen und die Applikation nutzen kann.
- **Als Benutzer** möchte ich mich mit meiner bei der Registrierung erstellten E-Mail-Adresse und meinem Passwort anmelden können, **damit ich** auf mein persönliches Benutzerkonto in der Applikation zugreifen kann.

3.4.2 Events

- **Als Benutzer** möchte ich Events erstellen können, **damit ich** eigene Veranstaltungen organisieren und veröffentlichen kann, die für alle Nutzer sichtbar sind.
- **Als Benutzer** möchte ich eine Übersichtsseite mit allen geplanten Events sehen können, **damit ich** schnell einen Überblick über bevorstehende Veranstaltungen erhalte und mich bei Interesse direkt dafür anmelden kann.
- **Als Benutzer** möchte ich einem Event beitreten können, **damit ich** meine Teilnahme an Veranstaltungen bestätigen und sicherstellen kann, dass ich für das Event registriert bin.
- **Als Benutzer** möchte ich ein Event, dem ich bereits beigetreten bin, wieder verlassen können, **damit ich** meine Teilnahme zurückziehen kann, falls ich doch nicht an der Veranstaltung teilnehmen kann.
- **Als Veranstalter** möchte ich Events, die ich erstellt habe, bearbeiten können, **damit ich** Informationen wie Titel, Datum oder Beschreibung aktualisieren und Änderungen an der Veranstaltung kommunizieren kann.
- **Als Veranstalter** möchte ich Events, die ich selbst erstellt habe, löschen können, **damit ich** fehlerhafte oder nicht mehr relevante Veranstaltungen aus dem System entfernen kann.

3.4.3 Wünsche

- **Als Benutzer** möchte ich einen Wunsch für eine Veranstaltung zu einem bestimmten Thema erstellen können, **damit ich** Vorschläge für neue Events einbringen kann.
- **Als Benutzer** möchte ich eigene erstellte Wünsche wieder löschen können, **damit ich** nicht mehr relevante oder falsch erstellte Wünsche entfernen kann
- **Als Benutzer** möchte ich Wünsche anderer Nutzer upvoten können, **damit ich** Interesse an vorgeschlagenen Veranstaltungen zeigen und unterstützen kann, dass diese umgesetzt werden.

- **Als Veranstalter** möchte ich mich zu einem Wunsch bereitstellen und daraus ein neues Event erstellen können, **damit ich** gezielt Veranstaltungen nach den Interessen der Nutzer anbieten kann.

3.4.4 Umfragen

- **Als Veranstalter** möchte ich innerhalb eines Events eine Umfrage für die Teilnehmer erstellen können, **damit ich** Feedback oder Meinungen zu bestimmten Themen einholen kann

- **Als Veranstalter** möchte ich eine von mir erstellte Umfrage bearbeiten können, **damit ich** Fragen oder Antwortoptionen anpassen kann, bevor die Umfrage abgeschlossen ist.

- **Als Veranstalter** möchte ich bei jedem Event die statistischen Auswertungen der Umfragen einsehen können, **damit ich** schnell einen Überblick darüber bekomme, zu welchen Optionen oder Themen die meisten Teilnehmer tendieren.

- **Als Benutzer** möchte ich einen Bereich haben, in dem ich alle meine noch nicht abgeschlossenen Umfragen sehen kann, **damit ich** den Überblick über offene Umfragen behalte und diese bei Bedarf weiterbearbeiten oder abschließen kann.

4 Funktionsumfang

Der Funktionsumfang vom Event-Planer wurde sorgfältig entwickelt, um eine umfassende und benutzerfreundliche Erfahrung für Mitarbeitende der Unternehmen pep.digital zu gewährleisten. Im Folgenden sind die Hauptfunktionen im Detail aufgeführt:

4.1 Anmeldung und Registrierung

Mitarbeitende können sich mühelos mit ihrem bestehenden Unternehmensaccount über Microsoft anmelden. Die benutzerfreundliche und intuitive Oberfläche optimiert den Anmeldeprozess. Zusätzlich haben auch externe Nutzer die Möglichkeit, sich unabhängig vom Unternehmen zu registrieren.

4.2 Events

User haben die Möglichkeit, eigene Events zu erstellen, zu bearbeiten und zu löschen. Jedes Event umfasst einen Titel und eine Beschreibung, in der Details zur Veranstaltung festgehalten werden können.

Nach der Erstellung können Events jederzeit vom Ersteller angepasst werden, sei es zur Änderung des Titels, der Beschreibung oder weiterer relevanter Informationen. Falls ein Event nicht mehr benötigt wird, kann es auch gelöscht werden.

Für eine bessere Übersicht gibt es einen **Event-Feed**, in dem alle Events angezeigt werden. Hier können User schnell durch die verschiedenen Veranstaltungen stöbern und sich inspirieren lassen.

Zudem gibt es eine eigene Sektion **„Meine Events“**, in der User ihre eigenen erstellten Events verwalten können. Hier können sie ihre Events ansehen, bearbeiten oder löschen.

Nach dem Event können individuelle Umfragen erstellt und jederzeit vom Ersteller bearbeitet werden. Diese Umfragen dienen dazu, Feedback und Meinungen der Teilnehmer zu sammeln. Mögliche Umfragethemen sind:

- **Terminfindung**: z.B. An welchen Freitagen im Monat können die meisten Teilnehmer teilnehmen?
- **Formatwahl**: Soll das Event digital oder in Präsenz stattfinden?
- **Sonstige Präferenzen**: Weitere individuelle Fragen, die der Ersteller festlegt.

Nach Abschluss des Events können die Teilnehmer das Event mit einer einfachen Sternebewertung (1-5 Sterne) bewerten. Diese Funktion hilft den Veranstaltern, schnell zu sehen, wie das Event bei den Teilnehmern ankam.

4.3 Wünsche

User haben die Möglichkeit, Wünsche zu erstellen, in denen sie äußern können, welche Events sie sich wünschen oder vorschlagen möchten. Jeder Wunsch enthält eine kurze Beschreibung des gewünschten Events.

Für eine bessere Übersicht gibt es einen Wünsche-Feed, in dem alle Wünsche angezeigt werden. Hier können User durch die verschiedenen Vorschläge stöbern und ihre Interessen zeigen.

Zudem gibt es eine eigene Sektion Meine Wünsche, in der User ihre eigenen erstellten Wünsche verwalten können. Hier können sie ihre Wünsche ansehen oder löschen.

Andere User können Wünsche upvoten, um zu zeigen, dass sie Interesse an diesem Event haben. Es gibt keine Mindestanzahl an Upvotes unabhängig von der Anzahl kann sich jemand bereitstellen, den Wunsch in ein tatsächliches Event umzusetzen.

Der Ersteller kann seinen Wunsch jederzeit bearbeiten oder löschen, falls sich die Idee ändert oder nicht mehr relevant ist.

Dieses Feature hilft dabei, Events zu organisieren, die wirklich von der Community gewünscht werden, und fördert die aktive Teilnahme aller User.

5 Aufwandsschätzung

5.1 Anforderungsanalyse

In dieser Phase werden die Anforderungen des Event-Planer festgelegt, einschließlich der Features und Funktionalitäten

- Dauer: 1 Woche

5.2 Entwurfsphase

Es wird das Design bzw. die Struktur der Anwendung erstellt, einschließlich der Benutzeroberfläche, der Datenbankstruktur und der Systemarchitektur.

- Dauer: 2 Woche

5.3 Implementierung & Testing

Entwicklung der Anwendung basierend auf den festgelegten Anforderungen und dem Design bzw. der Struktur. Applikation soll direkt nach jedem Feature auf Funktion getestet werden

- Dauer: 10 Wochen
  - Einarbeitung Technologien: 2 Wochen
  - Implementierung & Testing: 8 Wochen

5.4 Bereitstellung und Projektabschluss

Bereitstellung der Anwendung nach erfolgreicher Implementierung & Testphase.

- Dauer: 1-2 Wochen

6 Projektmanagement: "Scrum angelehnt"

Für dieses Projekt werden gewisse Apekte der agilen Methode Scrum verwendet. Dabei werden die Scrum-Zyklen auch bekannt als Sprints in 1 Wochen abschnitten eingeteilt, die sich durch schrittweise Entwicklung und durch regelmäßige Feedbackschleifen auszeichnet. Alle Aufgaben eines Sprints werden in Jira in einem Backlog gespeichert, beschrieben und unter dem Team verteilt.

Das Team hat wöchentliche Meetings mit dem Kunden und danach mit dem Betreuer des Projektes auch bekannt als Spint-Review. Dieser Termin findet immer Montags statt und markiert das Ende des aktuellen Sprints. Dieser Termin wird genutzt um die Ergebnisse des Kundes vom Sprint zu präsentieren und Feedback einzuholen, danach werden die nächsten Anforderungen vom Kunden besprochen. Im Anschluss darauf findet das Treffen mit dem Betreuer statt, bei denen offene fachliche Fragen geklärt werden.

Die Sprints beginnen immer am Montag nachdem alle Anforderungen und Feedbacks in dem Sprint-Review eingesammelt worden sind. Des weiteren wurde der Donnerstag und Sonntag als teaminternen Tag gekennzeichnet, bei dem weitere Fragen und Vorgehensweisen besprochen werden. Zudem werden die bearbeiteten Aufgaben durchgegangen und geklärt, was mit dem Kunden am Folgetag gesprochen wird. Nach dem Kundentreffen ist ein teaminternes Meeting angesetzt, bei welchem reflektiert wird, wie der letzte Sprint lief und mögliche Verbesserungen angesprochen werden.

Im Ablauf der Sprints sind zwei Daily Standup-Meetings pro Woche vorgesehen, da es aufgrund von Vorlesungen und anderen Laborveranstaltungen nicht möglich ist, sich täglich zu treffen. Dennoch sind die Teammitglieder während der gesamten Woche erreichbar, um Unterstützung zu bieten und bei Bedarf spontane Treffen zu organisieren. Für die interne Kommunikation wird ein Discord-Server verwendet, während für Besprechungen mit dem Betreuer und Kunden Microsoft Teams zum Einsatz kommt.

7 Systemarchitektur

7.1 Allgemein

Die Architektur des Gesamtsystems ist in einer Full-Stack Komponente unterteilt. Die Datenbank und der Authentifizierungsservice erfolgt über Supabase und einem Provider im Falle des Projektes über Microsoft, dass auch SSO unterstützt.

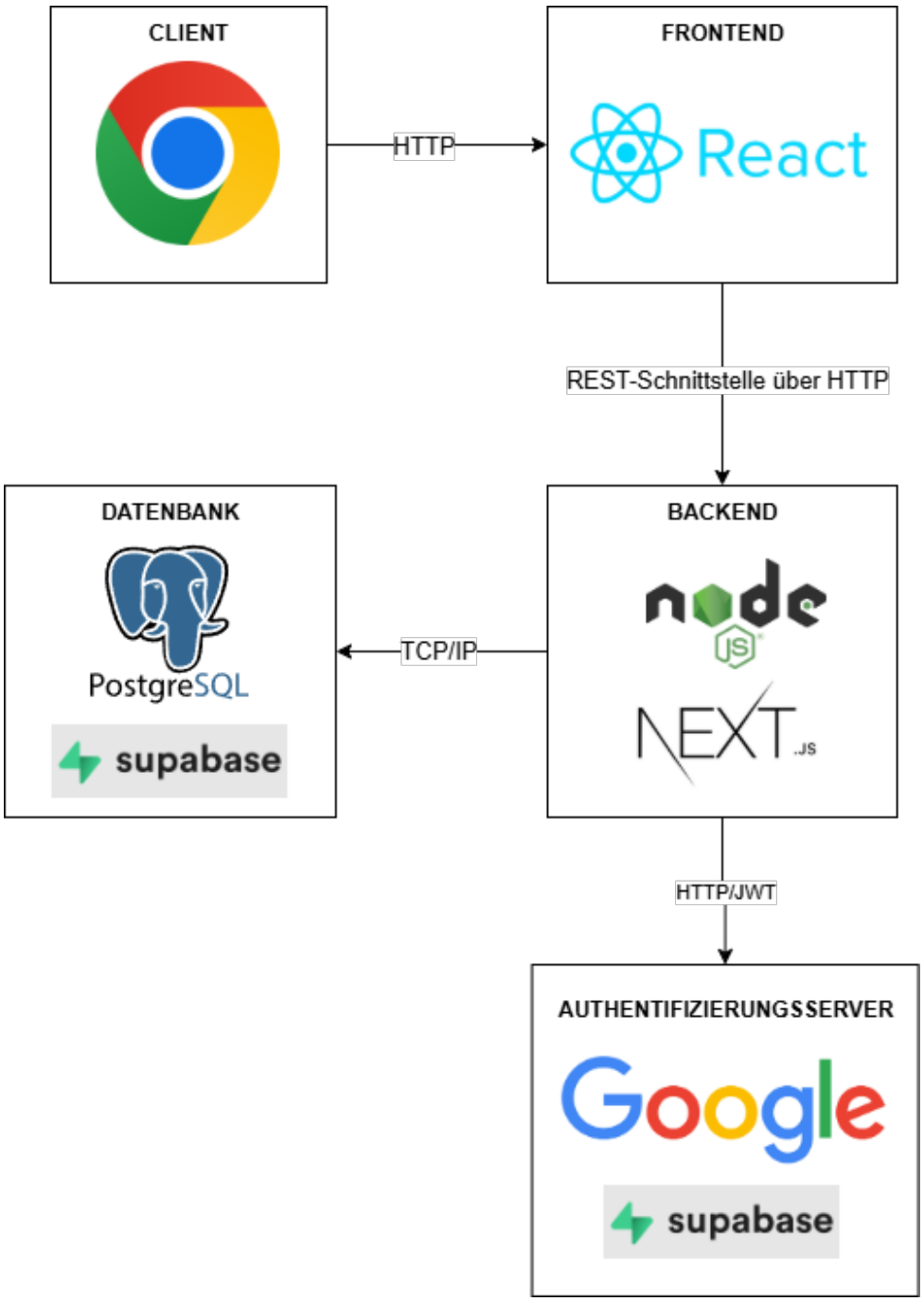

Abbildung 2: Systemarchitektur

7.2 Struktursicht

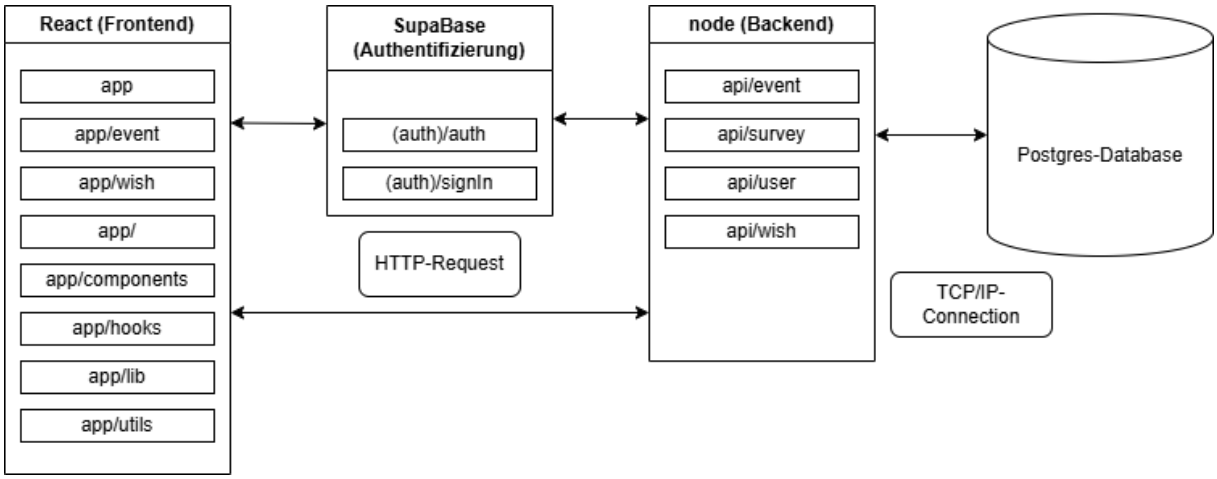

Abbildung 3: Struktursicht

Die Struktursicht zeigt den Aufbau des Projekts, das mit Next.js als Fullstack-Framework umgesetzt wurde. Die Anwendung ist in drei Hauptbereiche unterteilt: Das Frontend basiert auf React und ist modular strukturiert. Momentan wird für die Authentifizierung ein SSO benutzt, mit dem Authentication-Provider Google. Die Backend-Logik ist in Node.js implementiert und über die integrierten API-Routen von Next.js (z.B. api/event, api/user,...) realisiert. Es wird eine TCP/IP Verbindung für die Komunikation mit der Postgres-Datenbank genutzt. Die Architektur trennt somit klar zwischen Benutzeroberfläche, Geschäftslogik und Datenhaltung.

7.3 Verhaltenssicht

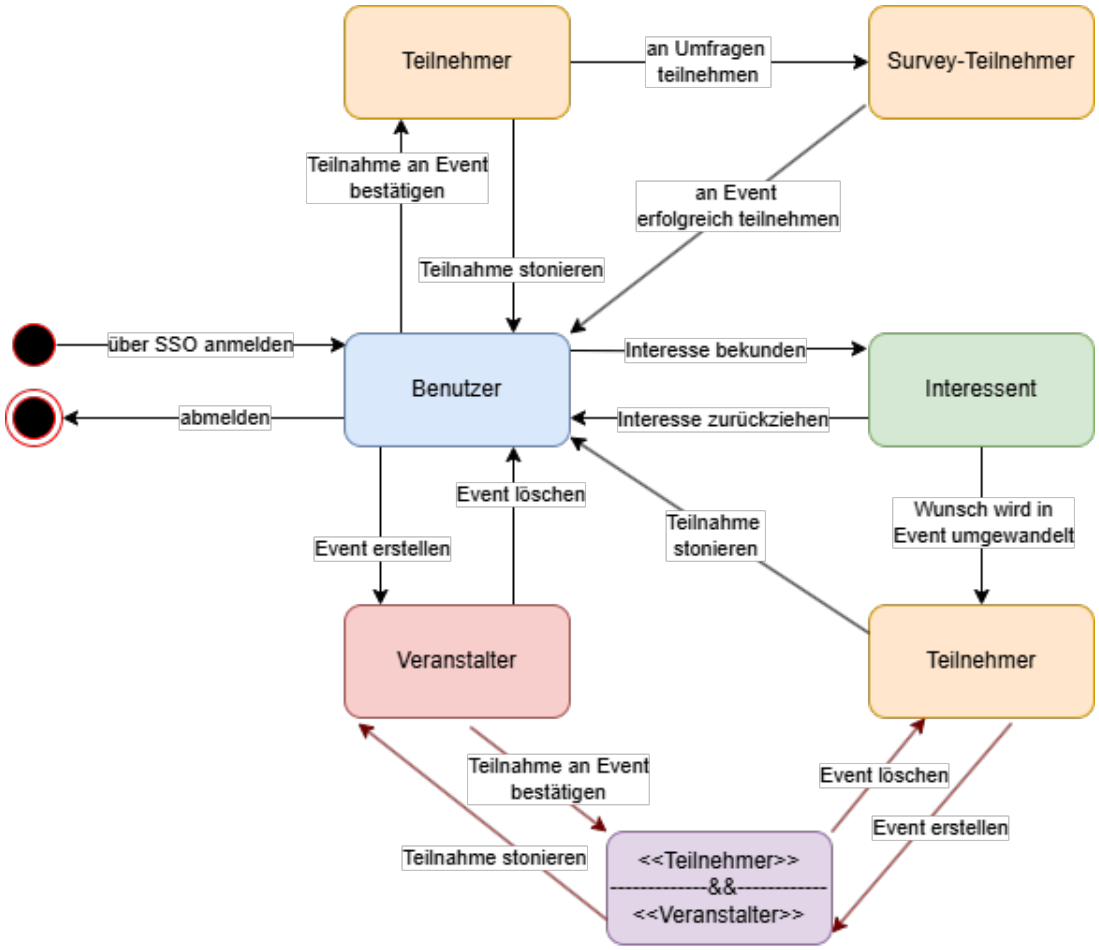

Abbildung 4: Verhaltenssicht

Die Verhaltenssicht beschreibt die Rollen und Interaktionen der Nutzer innerhalb der Anwendung. Ausgangspunkt ist der Benutzer, der sich über SSO anmelden und verschiedene Rollen einnehmen kann. Als Interessent kann er Interesse an einem Event bekunden oder zurückziehen. Sobald Interesse in eine Teilnahme übergeht, wird er zum Teilnehmer und kann Events bestätigen, stornieren oder an Umfragen teilnehmen. Wird ein Benutzer selbst aktiv, kann er als Veranstalter Events erstellen und verwalten. Die Grafik veranschaulicht diese dynamischen Zustandswechsel und Aktionen im System.

7.4 Verteilungssicht

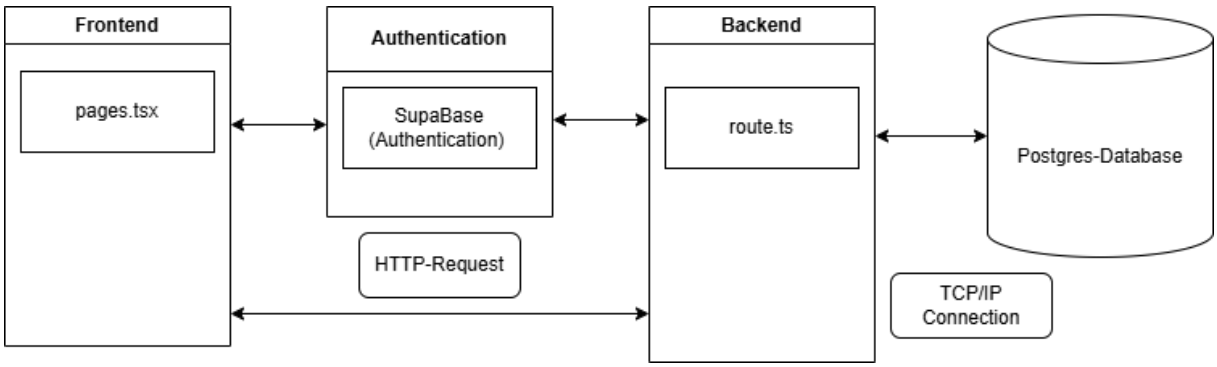

Abbildung 5: Verteilungssicht

Die Verteilungssicht zeigt, wie die verschiedenen Teile der Anwendung auf technische Komponenten verteilt sind. Das Frontend besteht aus React-Komponenten, die in pages.tsx implementiert sind. Die Authentifizierung erfolgt über SupaBase, das direkt vom Frontend per HTTP-Request angesprochen wird. Das Backend nutzt eine Datei wie route.ts, um serverseitige Logik bereitzustellen, und kommuniziert ebenfalls per HTTP mit der SupaBase-Datenbank. Die Datenhaltung sowie Authentifizierung sind somit ausgelagert und zentral über SupaBase realisiert. Die Struktur ermöglicht eine klare Trennung zwischen Präsentation, Authentifizierung, Logik und Daten.

8 Schnittstellentechnologien

8.1 Technische Umsetzung der Schnittstelle

Schnittstellentechnologien bilden das Fundament für die reibungslose Kommunikation zwischen den verschiedenen Komponenten unseres Softwaresystems. Ziel ist es, den Datenaustausch zwischen Frontend, Backend und Authentifizierungsmechanismen klar zu strukturieren undzuverlässig zu gestalten. Für unser Projekt haben wir uns bewusst für eine RESTful-API entschieden, um eine klar strukturierte und standardisierte Schnittstelle bereitzustellen. Die zentrale Aufgabe unseres Systems besteht darin, Events und Event-Wünsche anzulegen, anzuzeigen, zu bearbeiten und zu löschen, typische Operationen also, die sich sehr gut durch das CRUD-Modell (Create, Read, Update, Delete) abbilden lassen. REST ist dabei nicht nur leicht verständlich, was die Entwicklung erleichtert, sondern hat auch die Zusammenarbeit im Team gefördert. Als Datenformat verwenden wir JSON, die Übertragung erfolgt über das HTTPS-Protokoll. Damit ist die Kommunikation plattformunabhängig, sicher und performant. Die Architektur der API orientiert sich am App-Router-Konzept von Next.js, wobei jede Route in einer eigenständigen Datei (z.B. app/api/.../route.ts) implementiert wird. Die Anbindung an die Datenbank erfolgt über Prisma, die Benutzerverwaltung übernimmt Supabase. Ein weiterer Vorteil unserer API ist die Zustandslosigkeit: Jede Anfrage bringt alle nötige Informationen mit, z.B. Auth-Daten per JWT-Cookie. Dadurch brauchten wir keine eigene SessionVerwaltung. Auch bei der URI-Struktur setzen wir auf klare Adressierbarkeit, so wie beim Abrufen aller Events über GET /api/Event oder dem Upvote via POST /api/wish/[wishId]/upvote.

In der folgenden Tabelle 1 sind exemplarisch die wichtigsten Endpunkte für Events dargestellt:

Methode	**Pfad**	**Beschreibung**	**Erfolgsstatus**	**Fehlerstatus**
GET	/api/event	Gibt eine Liste aller Events zurück	200 OK	500 Internal Server Error
POST	/api/event	Erstellt ein neues Event	201 Created	400 Bad Request
GET	/api/event/[id]	Gibt ein einzelnes Event mit zugehörigen Daten zurück	200 OK	500 Internal Server Error
PATCH	/api/event/[id]	Aktualisiert gezielt Felder eines Events inkl. Terminen	200 OK oder 201 Created	400 Bad Request, 404 Not Found
DELETE	/api/event/[id]	Löscht ein Event	200 OK oder 204 No Content	404 Not Found

Tabelle 1: Übersicht der REST-API-Endpunkte für Events

PUT ist aktuell nicht implementiert, da im Projekt nur gezielte Änderungen nötig waren – ein vollständiges Überschreiben per PUT war bisher nicht erforderlich. Die verwendeten HTTP-Methoden entsprechen dem CRUD-Paradigma: Leseoperationen werden durch GET realisiert, das Anlegen neuer Ressourcen oder das Auslösen von Aktionen erfolgt mittels POST, während PATCH für gezielte Aktualisierungen und DELETE für das Entfernen von Ressourcen zuständig ist

9 Datenbank

9.1 Logisches Datenmodell

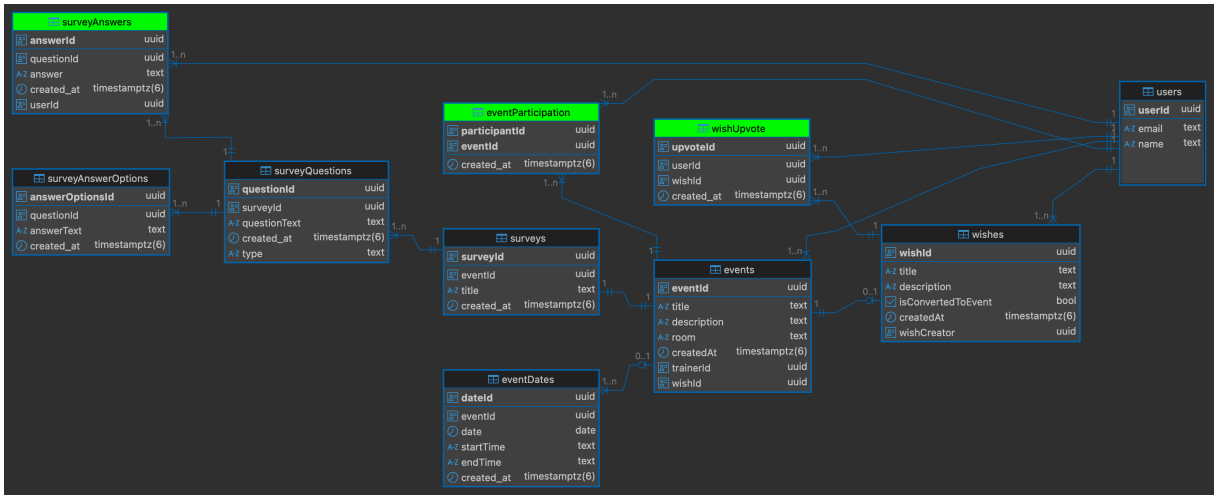

Abbildung 6: Logisches Datenmodell

9.1.1 Entity Relationship Modell

Die Anwendung verwendet eine relationale Datenbank auf Basis von PostgreSQL. Die Entscheidung fiel bewusst auf PostgreSQL, da es sich um ein bewährtes, leistungsstarkes Open-Source-Datenbanksystem handelt, das ACID-Konformität, flexible Abfragemöglichkeiten sowie gute Skalierbarkeit bietet – alles zentrale Anforderungen für das Eventmanagement, Umfrageverarbeitung und die Wunsch-Einreichung in der Anwendung. Zur Verwaltung und Bereitstellung der Datenbank wird Supabase verwendet, ein Backend-as-a-Service-Framework, das PostgreSQL als Kerntechnologie nutzt. Supabase bietet zusätzlich integrierte Funktionen wie Supabase Auth für die Authentifizierung, das in der Anwendung für die gesamte Authentifizierung und Nutzerverwaltung verwendet wird. Dadurch lässt sich der Entwicklungsaufwand im Backend deutlich reduzieren, ohne auf Sicherheit und Konsistenz bei der Zugriffskontrolle verzichten zu müssen.

9.1.2 Nutzerverwaltung

Ein zentraler Bestandteil dieser Architektur ist die enge Verknüpfung zwischen der eigens entwickelten Users-Tabelle und der von Supabase bereitgestellten Tabelle auth.users. Während sicherheitskritische Informationen wie Passwörter ausschließlich in auth.users abgelegt und geschützt bleiben, enthält die eigene Users-Tabelle ergänzende Angaben wie E-Mail-Adresse oder Anzeigename. Die Verbindung beider Tabellen erfolgt über das id-Feld, das in auth.users als Primary Key definiert ist. Ein automatisch ausgelöster Trigger sorgt dafür, dass diese ID unmittelbar nach der Registrierung auch in der Users-Tabelle als Foreign Key gespeichert wird. Dadurch wird jeder Nutzer eindeutig identifizierbar und eine klare Trennung zwischen sicherheitsrelevanten und anwendungsbezogenen Informationen geschaffen.

9.1.3 Haupttabellen und ihre Aufgabenbereiche

Die Datenbankstruktur gliedert sich in Haupt- und Hilfstabellen (vgl. Abbildung 6). Die Haupttabellen sind in der Abbildung schwarz dargestellt, während die Hilfstabellen farblich hervorgehoben (grün) sind. Zu den Haupttabellen zählen neben Users auch Wishes, Events sowie Surveys und SurveyQuestions. Über Wishes lassen sich Vorschläge - etwa für künftige Schulungsveranstaltungen - einreichen. Diese Vorschläge können von anderen Nutzern geupvoted oder in tatsächliche Events umgewandelt werden. Solche Events können jedoch auch unabhängig von Wünschen erstellt werden. Surveys ermöglichen es, Rückmeldungen einzuholen, beispielsweise zur Qualität von Events, und sind direkt mit den dazugehörigen SurveyQuestions verknüpft, die die konkreten Fragestellungen enthalten.

9.1.4 Hilfstabellen für komplexe Beziehungen

Die Hilfstabellen dienen vor allem der Abbildung komplexer, meist Viele-zu-Viele-Beziehungen. So wird in UserAnswer gespeichert, welche Antwort ein bestimmter Nutzer auf eine Umfragefrage gegeben hat. Die Tabelle EventParticipation dokumentiert, welche Nutzer sich für welche Events angemeldet haben. WishUpvote wiederum ermöglicht es, Schulungswünsche zu unterstützen, indem man signalisiert, dass man die Idee für sinnvoll hält und sich eine Umsetzung in Form eines Events wünscht.

9.2 Physisches Datenmodell

Das zuvor beschriebene logische Datenmodell wurde in PostgreSQL vollständig umgesetzt. Dabei wurde insbesondere auf Datenintegrität, Normalisierung und den gezielten Einsatz eines Triggers zur Pflege der eigenen Users-Tabelle geachtet. Primärschlüssel werden durchgängig als uuid realisiert, um eine systemübergreifend eindeutige Identifikation von Datensätzen zu gewährleisten. Alle Beziehungen zwischen den Tabellen sind über Foreign-Key-Constraints abgesichert, sodass beispielsweise ein Nutzer nur dann an einem Event teilnehmen oder eine Umfrage beantworten kann, wenn er zuvor auch für das entsprechende Event registriert wurde. Die referenzielle Integrität wird durch entsprechende Foreign-Key-Constraints auf Datenbankebene gewährleistet. Die Datenbankstruktur wurde in der dritten Normalform (3NF) gehalten. Im Datenmodell ist dies erkennbar an der konsequenten Trennung von Entitäten wie Users, Wishes, Events und Surveys, die jeweils klar abgegrenzte Aufgabenbereiche und Attribute besitzen. Jede Tabelle enthält ausschließlich Attribute, die direkt vom Primärschlüssel abhängig sind. Es gibt keine transitive Abhängigkeit zwischen Nicht-Schlüsselattributen. Beispielsweise sind die Antworten auf Umfragen klar normalisiert: Die gewählten Optionen (SurveyAnswerOptions) und die tatsächlichen Nutzerantworten (SurveyAnswers) sind strikt getrennt, inklusive eindeutiger Referenzen auf Fragen und Nutzer. Durch diese Struktur wird Redundanz vermieden. Die Tabelle Users ist über das Feld userId mit der Supabase-Authentifizierungstabelle auth.users verknüpft. Wie in Abschnitt 9.1.2 beschrieben, stellt ein in PostgreSQL implementierter Trigger sicher, dass beim Anlegen eines neuen Auth-Users automatisch ein entsprechender Eintrag in der eigenen Users-Tabelle erzeugt wird. Dies gewährleistet eine saubere Trennung sicherheitsrelevanter Informationen von anwendungsspezifischen Profildaten.

A UI Entwürfe

Anmerkung: Die UI-Entwürfe sind keine endgültigen Designentscheidungen, sondern dienen lediglich zur Veranschaulichung der Anwendungsstruktur.

A.1 Login

Abbildung 7: Login

A.2 Events

A.2.1 Event-Feed

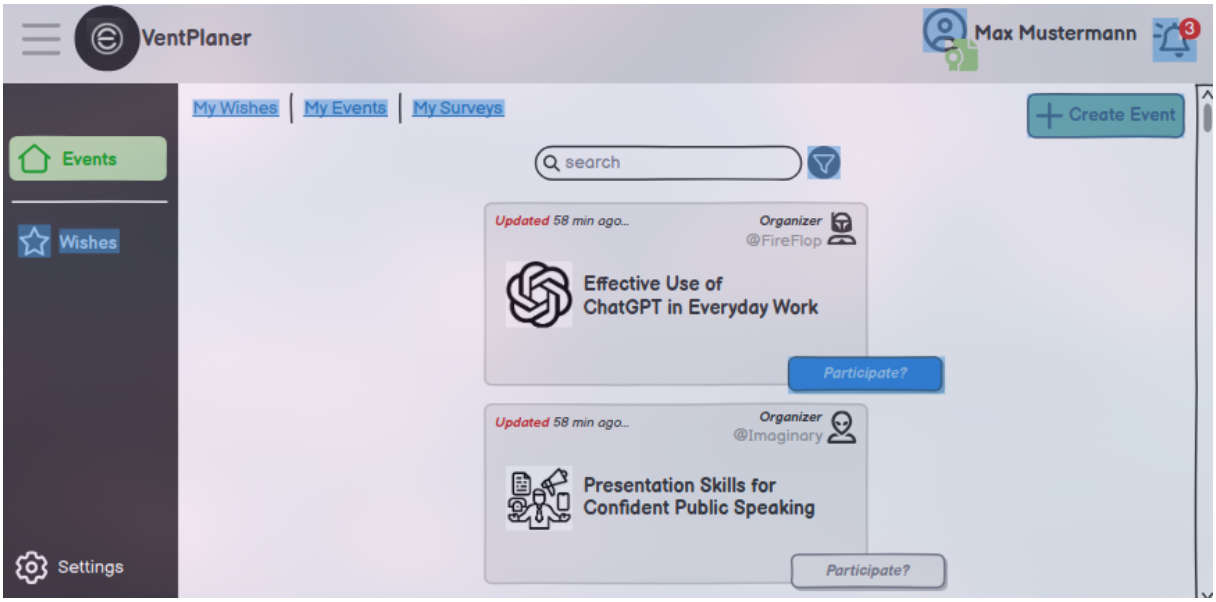

Abbildung 8: Event-Feed

A.2.2 Meine Events

Abbildung 9: Meine Events

A.2.3 Event erstellen

Abbildung 10: Event erstellen

A.2.4 Umfrage erstellen

Abbildung 11: Umfrage erstellen

A.2.5 Umfrage bearbeiten

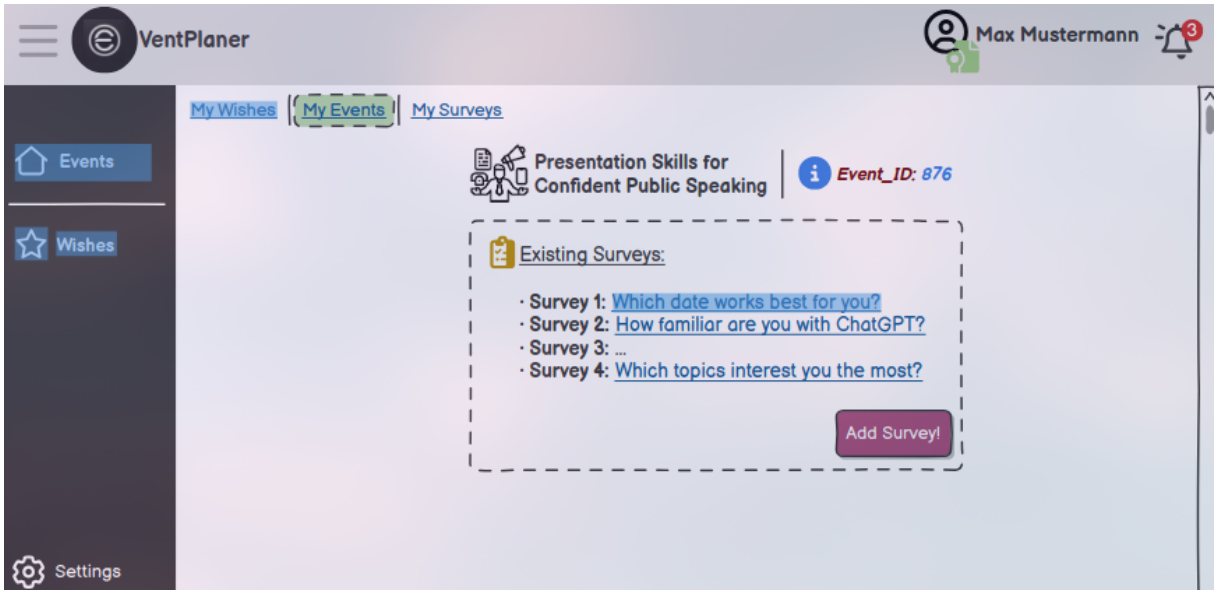

Abbildung 12: Umfrage bearbeiten

A.2.6 Meine Umfragen

Abbildung 13: Meine Umfragen

A.2.7 Event bearbeiten

Abbildung 14: Event bearbeiten

A.2.8 Event löschen

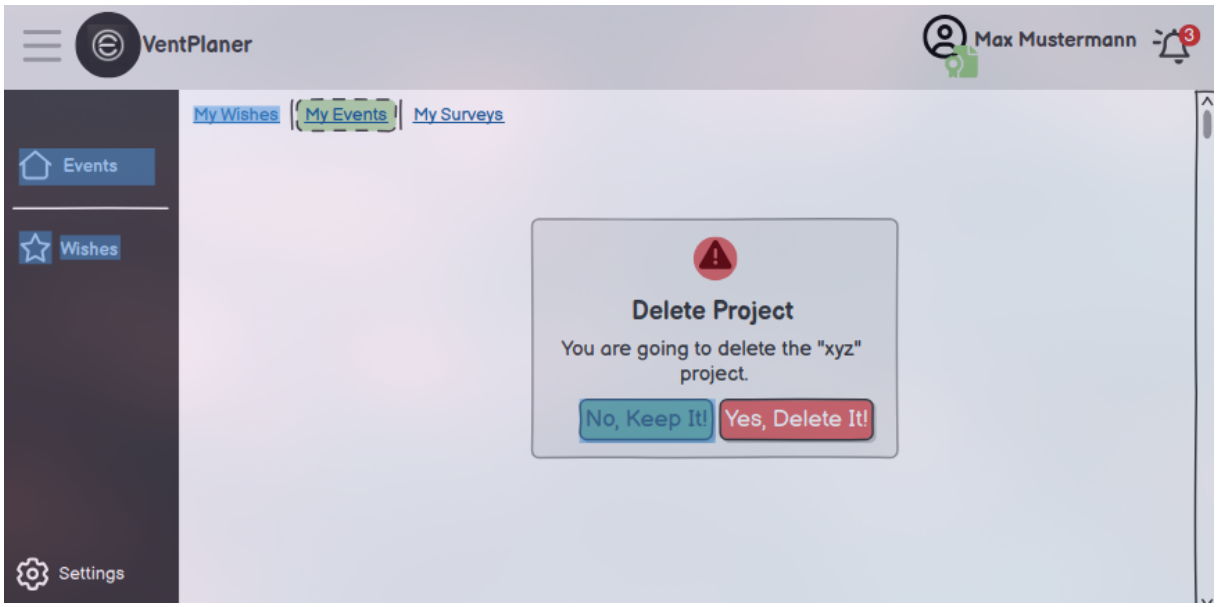

Abbildung 15: Event löschen

A.3 Wünsche

A.3.1 Wünsche-Feed

Abbildung 16: Wünsche-Feed

A.3.2 Meine Wünsche

Abbildung 17: Meine Wünsche

A.3.3 Wunsch erstellen

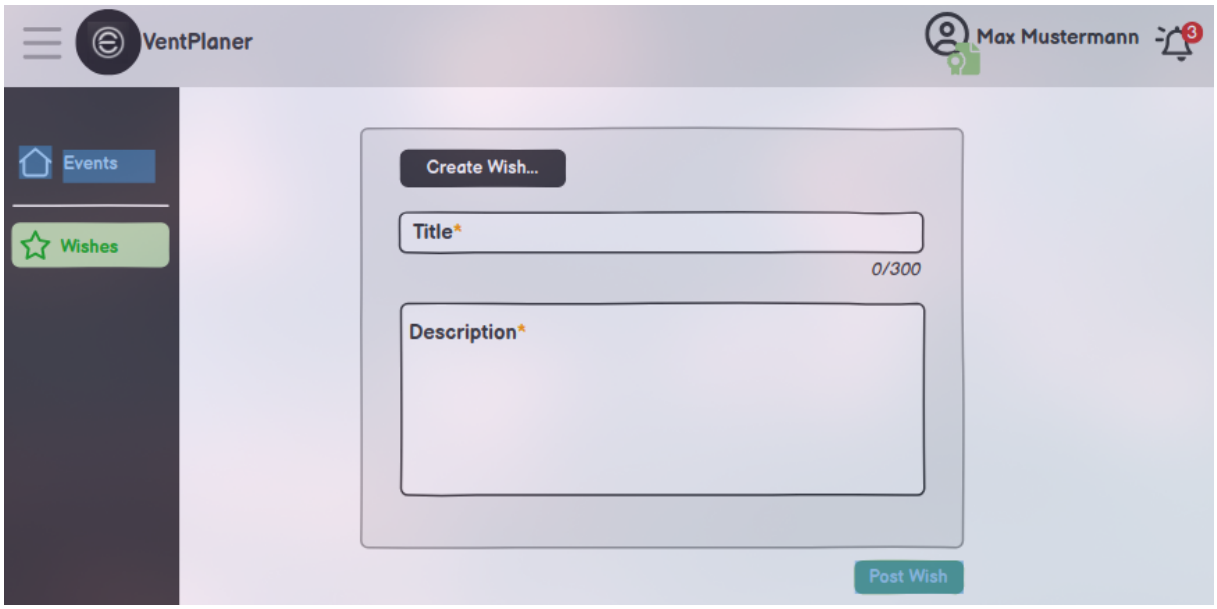

Abbildung 18: Wunsch erstellen

B Literaturverzeichnis