Event-Planer

Webapplikation zur Planung und Durchführung interner Events

Abbildung 1: Logo des Event-Planers
Quelle: `https://chatgpt.com/s/m_685af32608c4819195bb875081578cd1`

Projektteam:	Felix Hoffmann Baran Bickici Sami Gökpinar Ergün Bickici
Partnerunternehmen:	pep.digital
Studiengang:	Softwaretechnik und Medieninformatik 4. Semester, Sommersemester 2025
Kunde:	Alexander Brendel
Betreuer:	Klemens Morbe
GitHub-Repository	`https://github.com/pndroa/event-planer`

24. Juni 2025

Inhaltsverzeichnis

1 Einführung **5**

2 Technologien **5**
2.1 Full-Stack Framework . . . 5
2.1.1 Frontend . . . 5
2.1.2 Backend . . . 5
2.2 Datenbank . . . 6
2.3 Authentifizierung . . . 6
2.4 Bereitstellung . . . 6

3 Anforderungsspezifikation **7**
3.1 Zielgruppe . . . 7
3.2 Kundenbefragung . . . 7
3.3 Feature-Map . . . 7
3.4 User Stories . . . 8
3.4.1 Anmeldung . . . 8
3.4.2 Events . . . 8
3.4.3 Wünsche . . . 8
3.4.4 Umfragen . . . 9

4 Funktionsumfang **10**
4.1 Anmeldung und Registrierung . . . 10
4.2 Events . . . 10
4.3 Wünsche . . . 11

5 Aufwandsschätzung **12**
5.1 Anforderungsanalyse . . . 12
5.2 Entwurfsphase . . . 12
5.3 Implementierung & Testing . . . 12
5.4 Bereitstellung und Projektabschluss . . . 12

6 Projektmanagement: "Scrum angelehnt" **13**

7 Systemarchitektur **14**
7.1 Allgemein . . . 14
7.2 Struktursicht . . . 15
7.3 Verhaltenssicht . . . 16
7.4 Verteilungssicht . . . 17

8 Schnittstellentechnologien **18**
8.1 Technische Umsetzung der Schnittstelle . . . 18

9 Datenbank **19**
9.1 Logisches Datenmodell . . . 19
9.1.1 Entity Relationship Modell . . . 19
9.1.2 Nutzerverwaltung . . . 19
9.1.3 Haupttabellen und ihre Aufgabenbereiche . . . 20
9.1.4 Hilfstabellen für komplexe Beziehungen . . . 20

9.2 Physisches Datenmodell . . . 20

10 Installations- & Administrationshandbuch **21**
10.1 Lokale Installation . . . 21
10.2 Containerisierte Ausführung mit Docker . . . 21
10.3 Projektstruktur . . . 21

11 Lizenzen **22**
11.1 Verwendete Lizenzen . . . 22
11.2 Lizenz des eigenen Quellcodes . . . 22

12 Ausblick **23**
12.1 Kalenderansicht / Kalendersystem . . . 23
12.2 Zeitlich gruppierter Eventseite . . . 24
12.3 Kommentarsektion . . . 25
12.4 Profilseite . . . 26
12.5 Wiederkehrende Events . . . 27

13 Aufgabenverteilung & Zusammenarbeit im Projekt **28**

14 Vergleich von Aufwandsschätzung & Realaufwand **30**

15 Reflexion & Lernfortschritt **32**
15.1 Technische Fähigkeiten . . . 32
15.2 Methodische Kompetenzen . . . 32
15.3 Teamarbeit und Kommunikation . . . 33
15.4 Soft Skills . . . 33
15.5 Lessons Learned . . . 33
15.6 Fazit . . . 34

16 Interaktion mit Künstlicher Intelligenz **35**

A UI Entwürfe **36**
A.1 Login . . . 36
A.2 Events . . . 36
A.2.1 Event-Feed . . . 36
A.2.2 Meine Events . . . 37
A.2.3 Event erstellen . . . 37
A.2.4 Umfrage erstellen . . . 38
A.2.5 Umfrage bearbeiten . . . 38
A.2.6 Meine Umfragen . . . 39
A.2.7 Event bearbeiten . . . 39
A.2.8 Event löschen . . . 40
A.3 Wünsche . . . 40
A.3.1 Wünsche-Feed . . . 40
A.3.2 Meine Wünsche . . . 41
A.3.3 Wunsch erstellen . . . 41

B Literaturverzeichnis **44**

Abbildungsverzeichnis

1 Logo des Event-Planers . . . 1
2 Feature-Map . . . 7
3 Systemarchitektur . . . 14
4 Struktursicht . . . 15
5 Verhaltenssicht . . . 16
6 Verteilungssicht . . . 17
7 Logisches Datenmodell . . . 19
8 Kalendersystem (KI-generiert mit ChatGPT-4o, OpenAI, 2025) . . . 23
9 Events gruppiert nach Zeitabschnitten (generiert mit ChatGPT-4o, OpenAI, 2025) . . . 24
10 Kommentarsektion (generiert mit ChatGPT-4o, OpenAI, 2025) . . . 25
11 Profilseite (generiert mit ChatGPT-4o, OpenAI, 2025) . . . 26
12 Wiederkehrende Events (generiert mit ChatGPT-4o, OpenAI, 2025) . . . 27
13 Prozentuale Verteilung der Jira-Tickets nach Hauptfeature . . . 29
14 Stacey Matrix . . . 31
15 Login . . . 36
16 Event-Feed . . . 36
17 Meine Events . . . 37
18 Event erstellen . . . 37
19 Umfrage erstellen . . . 38
20 Umfrage bearbeiten . . . 38
21 Meine Umfragen . . . 39
22 Event bearbeiten . . . 39
23 Event löschen . . . 40
24 Wünsche-Feed . . . 40
25 Meine Wünsche . . . 41
26 Wunsch erstellen . . . 41

Tabellenverzeichnis

1 Exemplarische Endpunkte der Event-API 18

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 as a Service 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 Anforderungsspezifikation

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 2: 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 3: Systemarchitektur

7.2 Struktursicht

Abbildung 4: 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 5: 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 6: 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 und zuverlässig 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 nötigen 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: Exemplarische Endpunkte der Event-API

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. Die aktuellen APIs sind im GitHub-Repository mittels einer OpenAPI-Swagger-Datei dokumentiert

9 Datenbank

9.1 Logisches Datenmodell

Abbildung 7: 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 7). 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.

10 Installations- & Administrationshandbuch

Die Installations- und Administrationsanleitung befindet sich vollständig dokumentiert in der Datei README.md im Hauptverzeichnis des GitHub-Repositories. Darin werden zwei Varianten zur Ausführung der Anwendung beschrieben: eine lokale Installation sowie eine containerisierte Ausführung mittels Docker.

10.1 Lokale Installation

Für die lokale Nutzung werden Node.js (Version 18 oder höher), ein Paketmanager (npm oder yarn) sowie eine PostgreSQL Datenbank die bei diesem Projekt über Supabase läuft vorausgesetzt. Die notwendigen Umgebungsvariablen sind in einer .env-Datei zu definieren, für die eine Vorlage in der Datei .env.template bereitgestellt wird.

Die Schritte zur Installation und Ausführung beinhalten unter anderem:

- Klonen des Repositories
- Installation der Abhängigkeiten
- Generierung des Prisma-Clients
- Start der Anwendung im Entwicklungs- oder Produktionsmodus

Zudem werden administrative Befehle für Prisma dokumentiert, etwa zur Migration oder zur Verwendung des webbasierten Datenbank-Interfaces.

10.2 Containerisierte Ausführung mit Docker

Alternativ kann die Anwendung vollständig containerisiert ausgeführt werden. Die dazu erforderlichen Konfigurationsdateien Dockerfile und docker-compose.yml sind im Projekt enthalten. Die Docker-Umgebung berücksichtigt unter anderem die Abhängigkeit von OpenSSL und die Nutzung von Supabase mit einem IPv4-kompatiblen Session Pooler.

Die README-Datei beschreibt den Aufbau, Start, Betrieb und das Stoppen der Containerumgebung sowie die Ausführung von Verwaltungsbefehlen innerhalb des Containers.

10.3 Projektstruktur

Ebenfalls in der README.md enthalten ist ein Überblick über die Projektstruktur, einschließlich Konfigurationsdateien, Verzeichnisse für Quellcode, Datenbankschemata und Containerdefinitionen.

11 Lizenzen

11.1 Verwendete Lizenzen

Im Projekt wird das Framework Next.js verwendet, das unter der MIT-Lizenz steht. Darüber hinaus kommen zahlreiche weitere Open-Source-Bibliotheken zum Einsatz, die überwiegend ebenfalls unter der MIT-Lizenz veröffentlicht wurden. Dazu gehören unter anderem:

- React und React-DOM (MIT)
- Prisma (MIT)
- MUI (Material UI) (MIT)
- Emotion (verschiedene Module wie styled, cache, react) (MIT)
- Axios (MIT)
- Date-fns und date-fns-tz (MIT)
- Supabase (MIT)
- EJS (Apache-2.0)
- Diverse ESLint- und Testing-Bibliotheken (überwiegend MIT)

Die genauen Lizenzinformationen aller verwendeten Pakete sind in der Datei package-lock.json hinterlegt. Dabei handelt es sich fast ausschließlich um weit verbreitete, permissive Lizenzen wie MIT, Apache-2.0 oder BSD.

11.2 Lizenz des eigenen Quellcodes

Der im Rahmen dieses Projekts erstellte Quellcode wird unter die MIT-Lizenz gestellt. Die Lizenz erlaubt eine freie Nutzung, Veränderung und Weiterverbreitung des Codes, auch für kommerzielle Zwecke, sofern die Lizenzbedingungen eingehalten werden. Der vollständige Lizenztext befindet sich in der Datei `LICENSE` im Hauptverzeichnis des Projekts.

12 Ausblick

Das bisherige Eventplaner erfüllt alle gewünschten Anforderungen und funktioniert im Alltag zuverlässig. Trotzdem ergeben sich beim Arbeiten mit der Anwendung viele Ideen, wie sie in Zukunft weiterentwickelt werden könnte

12.1 Kalenderansicht / Kalendersystem

Momentan werden geplante Events in einer einfachen Feed-Liste angezeigt. Für viele Nutzer könnte es hilfreicher sein, diese Veranstaltungen zusätzlich in einem Kalender zu sehen. Eine klassische Monats- oder Wochenansicht kann einen besseren Überblick über anstehende Termine vermittelt. Durch einen solchen Kalender würde erkennbar, wie stark bestimmte Tage belegt sind oder wo noch Platz für weitere Veranstaltungen bleibt. Auch Terminüberschneidungen oder Lücken im Plan würden sichtbar. Gerade wenn mehrere Events organisiert oder besucht werden, könnte diese visuelle Darstellung dabei helfen, schneller Entscheidungen zu treffen. Für den kann Kunden kann so ein Mehrwert entstehen, denn die Anwendung wirkt strukturierter und lässt sich flexibler im Alltag einsetzen.

Abbildung 8: Kalendersystem (KI-generiert mit ChatGPT-4o, OpenAI, 2025)

12.2 Zeitlich gruppierter Eventseite

Aktuell werden die Events einfach untereinander angezeigt, egal ob sie heute stattfinden oder erst in ein paar Wochen. Das ist zwar funktional, aber mit der Zeit wird es unübersichtlich. Eine Gruppierung nach Zeitabschnitten könnte helfen.

Die Idee: Events werden automatisch nach ihrem Startdatum sortiert und in logische Abschnitte gegliedert, zum Beispiel **„Heute“**, **„Diese Woche“**, **„Nächste Woche“** oder auch einfach mit einem Datumstitel. So erkennt man auf einen Blick, was demnächst ansteht, ohne einzeln in der EventDetail-Seite suchen zu müssen. Das könnte praktisch für Vielnutzer oder bei größeren Organisationen mit zahlreichen Events werden.

Abbildung 9: Events gruppiert nach Zeitabschnitten (generiert mit ChatGPT-4o, OpenAI, 2025)

12.3 Kommentarsektion

Obwohl alle grundlegenden Informationen zu einem Event direkt auf der Detailseite ersichtlich sind und sogar eigene Umfragen erstellt werden können, fehlt bisher eine einfache Möglichkeit, sich spontan oder informell auszutauschen.

Ein Kommentarbereich würde genau hier ansetzen. Teilnehmer könnten sich dort kurzschließen, wenn jemand etwas mitbringen möchte. Auch organisatorische Hinweise, die nicht direkt in die Eventbeschreibung gehören, hätten dort ihren Platz

So entsteht Raum für lockere Kommunikation direkt am Ort. Das könnte die Plattform persönlicher und interaktiver machen.

Abbildung 10: Kommentarsektion (generiert mit ChatGPT-4o, OpenAI, 2025)

12.4 Profilseite

Aktuell gibt es in der Anwendung keine Möglichkeit, das eigene Profil einzusehen oder zu bearbeiten. Wer eingeloggt ist, sieht weder die eigene Mailadresse noch frühere Aktivitäten oder Events, an denen man teilnimmt. Auch andere Nutzerprofile sind nicht erreichbar.

Eine persönliche Profilseite würde genau hier ansetzen. Nutzer könnten dort ihre bisherigen Events einsehen, z. B. als Teilnehmer oder Organisator. Auch grundlegende Profilinformationen wie Name, E-Mail oder ein Avatar ließen sich dort darstellen und eventuell bearbeiten.

Das könnte nicht nur die Nutzerbindung verbessern, sondern auch Orientierung schaffen, wer hinter welchen Events oder Ideen steckt.

Abbildung 11: Profilseite (generiert mit ChatGPT-4o, OpenAI, 2025)

12.5 Wiederkehrende Events

Im System gibt es viele wiederkehrende Formate wie interne Schulungen oder Vorträge. Momentan müssen sie jedes Mal neu erstellt werden, auch wenn sich nur das Datum ändert. Es könnte den Komfort verbessern, wenn man beim Anlegen eines Events direkt angeben könnte, dass es sich wiederholt. Ob wöchentlich, monatlich oder nach einem benutzerdefinierten Rhythmus. Das System würde dann automatisch eine Event-Serie anlegen. Die Inhalte bleiben dabei gleich, nur die Termine unterscheiden sich. Gerade bei Formaten wie internen Schulungen spart das viel Aufwand und sorgt dafür, dass keine Termine vergessen werden. Auch für Teilnehmende kann es transparenter werden, wann die nächsten Sessions stattfinden.

Abbildung 12: Wiederkehrende Events (generiert mit ChatGPT-4o, OpenAI, 2025)

13 Aufgabenverteilung & Zusammenarbeit im Projekt

Im Team Eventplaner hat grundsätzlich jede Person sowohl Frontend- als auch Backend-Aufgaben übernommen. Trotzdem haben sich im Laufe des Projekts natürliche Schwerpunkte gebildet, bei denen einzelne Mitglieder bestimmte Features federführend umgesetzt haben.

Ergün war technischer Ansprechpartner, insbesondere im Bereich Authentifizierung und Supabase. Er hat die Anmeldung und Registrierung umgesetzt und dafür gesorgt, dass die Datenbank angebunden war. Außerdem hat er das Deployment über Vercel übernommen und sich um technische Aspekte gekümmert, die beispielsweise mit der Anbindung an die Cloud zusammenhingen. Zu Beginn des Projekts kam es aufgrund der Server-Region von Supabase, die zunächst in den USA lag, zu Latenzproblemen. Ergün hat in diesem Zusammenhang die Umstellung auf einen Server-Standort in Frankfurt organisiert. Zusätzlich hat er das Survey-Feature umgesetzt, das im Backend zusätzliche Logik erforderte. Auch bei Git-Problemen oder MergeKonflikten hat er häufig die Lösung übernommen.

Felix hat sich mit der E-Mail-Funktion beschäftigt, also mit allen Aspekten, die dafür sorgen, dass Teilnehmende bei bestimmten Aktionen automatisch benachrichtigt werden. Die Umsetzung erforderte eine enge Abstimmung zwischen Backend und Frontend. Darüber hinaus hat er beim Refactoring mitgewirkt, den Code besser strukturiert und auf Übersichtlichkeit geachtet.

Baran war vor bei der Datenbank und der Logik für Event-Teilnahmen aktiv. Er hat zu Beginn die Datenbank modelliert und diese anschließend fortlaufend an die Anforderungen angepasst. Zusätzlich hat er mit Balsamiq eine erste visuelle Version der App erstellt, die in der weiteren Entwicklung genutzt wurde. Außerdem hat er Backend-Logik rund um Event-Teilnahmen sowie zusammen mit Sami das Statistik Feature umgesetzt.

Sami hat sich mit dem Upvote-Feature beschäftigt, mit dem Nutzer Wishes bewerten können. Außerdem hat er die Logik für In-App-Benachrichtigungen umgesetzt, also der Logik dahinter, wann Nutzer welche Informationen bekommen sollen, sowie der dazugehörigen Funktionalität. Ein weiterer Schwerpunkt war auch das Statistik Feature, bei dem er eng mit Baran zusammengearbeitet hat. Zusätzlich hat Sami das grundlegende Layout der App erstellt, einschließlich Navigation, Header und Footer, wodurch eine Struktur für die weitere Entwicklung entstand. Die Zusammenarbeit im Team verlief insgesamt strukturiert, auch wenn es Phasen gab, in denen einzelne Mitglieder überwiegend an spezifischen Features gearbeitet haben, wurden bei Problemen oder größeren Herausforderungen gemeinsame Lösungen erarbeitet. Die Arbeitsverteilung war ausgewogen, und alle Teammitglieder haben im Verlauf des Projekts sowohl im Backend als auch im Frontend Erfahrungen gesammelt.

Abbildung 13: Prozentuale Verteilung der Jira-Tickets nach Hauptfeature

Die Grafik zeigt die prozentualen Beiträge der einzelnen Teammitglieder zu den jeweiligen Hauptfeatures, basierend auf den erfassten Jira-Tickets. Dabei ist wichtig zu beachten, dass die dargestellten Prozentwerte den Anteil an der Gesamtarbeit innerhalb eines spezifischen Features widerspiegeln, nicht jedoch den Umfang oder Schwierigkeitsgrad der jeweiligen Funktion selbst

Nicht alle Features sind gleich umfangreich oder komplex. Manche Bereiche, wie zum Beispiel das Statisticsoder Survey-Feature, bringen zusätzlichen technischen Aufwand mit sich, während andere eher frontendlastig sind oder sich stärker auf Design und Struktur konzentrieren. Die reine Aufteilung nach Ticket-Anteilen liefert daher nur einen groben Überblick, wer sich inhaltlich bei welchem Bereich eingebracht hat. Sie lässt jedoch keine direkten Rückschlüsse auf den tatsächlichen Zeit- oder Arbeitsaufwand zu.

Die Visualisierung verdeutlicht dennoch die im Projekt entstandenen Schwerpunkte und bestätigt, dass einzelne Teammitglieder bestimmte Features federführend umgesetzt haben. Gleichzeitig zeigt sich auch, dass es übergreifende Zusammenarbeit gab, bei der alle Mitglieder in verschiedenen Bereichen mitgewirkt und sowohl Backend- als auch Frontend-Erfahrungen gesammelt haben.

14 Vergleich von Aufwandsschätzung & Realaufwand

Im Kurs wird von einer ungefähren wöchentlichen Arbeitszeit von 16 Stunden ausgegangen. Zu Beginn des Projekts wurde der Zeitaufwand für die Planung der einzelnen Sprints unterschätzt. Das Erstellen vollständiger und für alle Entwickler verständlicher User-Stories benötigte regelmäßig bis zu zwei Stunden. Auch die Besprechungen mussten im Schnitt mit bis zu zwei Stunden bemessen werden. So nahmen die Sprintplanung und durchführung, unserer abstrahierten Version von Scrum, insgesamt bis zu fünf Stunden der wöchentlich angesetzten Arbeitszeit ein.

Unabhängig von der wöchentlichen Arbeitszeit hat das Erstellen einer stabilen Datenbankstruktur für den Event-Planer länger gedauert als ursprünglich erwartet. Die Planung der Datenbankstruktur nahm zu Beginn des Projekts neben dem Erstellen des Designs der Applikation einen großen Teil der Arbeitszeit ein. Die gesamte Zeit belief sich wie geschätzt auf knapp zwei Wochen. Jedoch wurden auch im Laufe des Projekts immer wieder Änderungen an der Datenbank vorgenommen, da unnötige Relationen oder Entitäten entfernt oder benötigte hinzugefügt wurden.

Dadurch zog sich die Entwicklung der Datenbank über nahezu das gesamte Projekt hinweg. Die Implementierung, also Umsetzung der einzelnen User-Stories, wurde von Beginn an als am zeitintensivsten eingeschätzt. Ein Großteil des Projektzeitraums (10 Wochen) wurde dafür und für die Tests eingeplant. Im Verlauf des Projekts stellte sich jedoch heraus, dass der tatsächliche Aufwand stark vom jeweiligen Thema abhing. Manche Aufgaben erwiesen sich wie erwartet als komplex, während andere deutlich einfacher umzusetzen waren. Oft, weil wir passende Technologien fanden, die uns einen Extraaufwand sparten. Beispielsweise konnte durch Prisma viel Zeit beim Erstellen und Ergänzen der Datenbankstruktur eingespart werden. Dabei kristallisierte sich heraus, dass das Einschätzen der Komplexität von Aufgaben eine Herausforderung ist. Die Einschätzungen fielen sehr unterschiedlich aus, typabhängig, mal eher vorsichtig, mal eher knapp Wöchentlich nahm die Bearbeitungszeit neben den organisatorischen Tätigkeiten rund um die Meetings den Rest der Woche ein, also mindestens 11 Stunden.

Die Dokumentation der gesamten Anwendung wurde bereits zu Beginn als aufwändig eingeschätzt, zum Beispiel das Dokumentieren der API-Schnittstellen. Im Verlauf des Projekts konnten wir jedoch durch die konsequente Nutzung von Jira, insbesondere durch das regelmäßige Erstellen von User-Stories und deren detaillierte Beschreibung, den dokumentarischen Aufwand besser bewältigen. Da User-Stories konsequent ausführlich ausformuliert und dokumentiert wurden, konnte auf diese Weise langfristig Zeit eingespart werden. Dennoch wurde auch der Aufwand für die Dokumentation der einzelnen Meilensteine als nicht unerheblich betrachtet. In der Woche vor den jeweiligen Abgaben ein Extra von drei Stunden.

Schlussendlich wurde der Aufwand für das Testen mit Unit-Test am stärksten unterschätzt. Durch die Ambition, am Ende eines jeden Sprints neue Features präsentieren zu können, war es nicht möglich, diese Features durch Unit-Tests zu prüfen. So wurde lediglich manuell im Frontend getestet. Diese Arbeit summierte sich über den gesamten Projektverlauf hinweg und wurde schließlich so umfangreich, dass sie bis heute nicht vollständig abgeschlossen werden konnte. Rückblickend war der Aufwand, ein Feature sauber und gründlich zu testen, der am meisten unterschätzte Aspekt des Projekts.

Das Projekt hat deutlich gemacht, wie essenziell Unit-Tests und Tests im Allgemeinen für die Stabilität einer Anwendung sind. Idealerweise sollten Tests vor oder spätestens während der Entwicklung neuer Features erstellt werden. In Abhängigkeit vom gewählten Entwicklungsansatz kann es sogar sinnvoll sein, Tests bereits vor Beginn der eigentlichen Implementierung zu formulieren. Auf diese Weise lassen sich viele Fehlerquellen frühzeitig erkennen und vermeiden. Auch die Belastung während der Entwicklungsphasen hätte durch eine frühzeitige Testabdeckung spürbar reduziert werden können. Zudem wurde deutlich, dass die Planung einzelner Sprints einen erheblichen Zeitaufwand darstellt.

Vor diesem Hintergrund lässt sich sagen, dass in Anbetracht der Zeit und des Umfangs für dieses Projekt ein linearer Ansatz, möglicherweise mit dem Ausformulieren eines Lastenhefts, effektiver gewesen wäre. Da so viel Organisationszeit für das Scrum verfahren gespart worden wäre. Da sich der Umfang des Projekts noch im unteren Teil am Ursprung der Stacey-Matrix einschätzen lässt.

Trotzdem war es richtig und gut, eine abgeänderte Form von Scrum als Projektorganisationsmethode zu nutzen, da Scrum ein weit verbreiteter Industriestandard ist und so wertvolle Einblicke in das Berufsleben gewonnen werden konnten.

Abbildung 14: Stacey Matrix

Quelle: `https://chatgpt.com/s/m_6859cf49fe1c8191b4c5b9ef57609591`

15 Reflexion & Lernfortschritt

Durch das Semesterprojekt, bei dem für ein reales Unternehmen eine praxisnahe Anwendung entwickelt wurde, konnten sowohl fachliche als auch persönliche Erfahrungen gesammelt werden. Im Verlauf des Semesters fanden regelmäßige Meetings mit dem Betreuer und dem Kunden statt, die dabei halfen, die Arbeit an den Anforderungen auszurichten und kontinuierlich weiterzuentwickeln.

15.1 Technische Fähigkeiten

Im technischen Bereich wurden umfangreiche neue Kenntnisse erworben. Technologien wie Next.js, Supabase und Prisma waren zu Beginn für die meisten Teammitglieder neu. Einzelne Personen verfügten bereits über erste Vorkenntnisse im Umgang mit Next.js. Dennoch war es erforderlich, sich intensiv mit diesen Tools auseinanderzusetzen und verschiedene Herausforderungen zu bewältigen. Im weiteren Verlauf des Projekts wurde der Umgang mit den Technologien vertrauter wodurch Funktionen wie Authentifizierung, automatisierte E-Mail-Benachrichtigungen und Datenbankmodellierungen umgesetzt werden konnten. Das Deployment über Vercel ermöglichte zudem praktische Einblicke in HostingPlattformen und produktive Umgebungen.

Ein zusätzlicher technischer Schwerpunkt lag im Umgang mit Git. Insbesondere zu Beginn traten häufig Merge-Konflikt auf. Trotz der Nutzung von Feature-Branch kam es zu Überschneidungen, die im Technische Fähigkeiten Vorfeld nicht immer absehbar waren. Teilweise arbeiteten mehrere Personen parallel an denselben Komponenten, was die Konflikte begünstigte. Um diese Probleme zu reduzieren, wurde der Fokus stärker auf technisches Planen und Abstimmung innerhalb des Teams gelegt. Die Anwendung systematischer Fehleranalysen unterstützte dabei den Entwicklungsprozess. Insbesondere durch Debugging konnten Fehler schrittweise erkannt und behoben werden.

Im Bereich Testing und Qualitätssicherung wurde erkannt, wie wichtig automatisierte Tests für die langfristige Stabilität sind. Während der Projektphase wurden überwiegend manuelle Tests durchgeführt, indem die Anwendung wiederholt durchgeklickt und Funktionen überprüft wurden. Dennoch traten weiterhin Fehler auf, die teilweise vom Betreuer oder dem Kunden entdeckt wurden. Im Rückblick zeigte sich, dass ein früherer und konsequenter Einsatz von automatisierten Tests, einschließlich Exploratory-Tests Unit-Tests und End-to-End-Test, dabei geholfen hätte, solche Fehler frühzeitig zu identifizieren.

15.2 Methodische Kompetenzen

Neben den technischen Aspekten lag der Fokus zunehmend darauf, das Projekt strukturiert und wartbar umzusetzen. Regelmäßige RefactoringProzesse und Code Reviews unterstützten dabei, den Quellcode übersichtlich zu halten. Im Bereich Deployment zeigte sich hingegen Verbesserungspotenzial. Eine strukturierte Deployment-Lösung wurde nicht etabliert. Neue Versionen wurden direkt über den Haupt-Branch veröffentlicht. Dadurch kam es vereinzelt dazu, dass bereits deployte Änderungen durch nachfolgende Merges im selben Sprint unbeabsichtigt überschrieben wurden.

15.3 Teamarbeit und Kommunikation

Die Zusammenarbeit im Team entwickelte sich im Verlauf des Projekts. Zu Beginn wurde häufig isoliert gearbeitet. Ein Beispiel dafür war die parallele Entwicklung einer identischen Komponente, die letztlich in vier unterschiedlichen Versionen vorlag. Im weiteren Verlauf zeigte sich, wie entscheidend regelmäßiger und offener Austausch ist. Die internen Meetings wurden strukturierter, Entscheidungen konnten gemeinsam getroffen und Probleme frühzeitig identifiziert werden. Auch die Kommunikation mit dem Kunden und dem Betreuer verbesserte sich, insbesondere im Hinblick auf das Verständnis von Anforderungen sowie den konstruktiven Umgang mit Rückmeldungen.

15.4 Soft Skills

Im Verlauf des Projekts wurden zudem Fähigkeiten im Bereich Zeitmanagement, Eigenverantwortung und Selbstorganisation weiterentwickelt. Die Planung von Arbeitsaufwänden und das realistische Einschätzen von Aufgaben erwiesen sich als notwendig, um gesetzte Termine einzuhalten. Flexibilität spielte ebenfalls eine wichtige Rolle, da nicht alle Vorhaben wie geplant umgesetzt werden konnten. Herausforderungen wurden durch alternative Lösungsansätze bewältigt, beispielsweise durch spontane Meetings oder gezielte Rückfragen innerhalb des Teams.

Zudem wurden Belastbarkeit und Ausdauer gestärkt. Insbesondere wiederkehrende Fehler oder technische Probleme machten es erforderlich, über einen längeren Zeitraum lösungsorientiert zu arbeiten. Teamarbeit und Kommunikation Soft Skills Diese Erfahrungen unterstützten den Umgang mit Rückschlägen und den gezielten Erwerb neuer Lösungsansätze.

15.5 Lessons Learned

Im Verlauf des Projekts wurden neben positiven Aspekten auch verschiedene Verbesserungspotenziale deutlich. Ein zentrales Thema war die fehlende Konsequenz bei der Integration automatisierter Tests. Die schnelle Umsetzung neuer Features hatte dabei häufig Priorität, wodurch das Testing zu spät oder nur unvollständig erfolgte. Rückblickend zeigte sich, dass die frühzeitige und regelmäßige Einbindung von Tests parallel zur Entwicklung sinnvoll gewesen wäre, um Fehlerquellen frühzeitig zu minimieren und aufwendige DebuggingSitzungen zu vermeiden.

Ein weiterer Aspekt war die Abhängigkeit von Supabase als externer Cloud-Dienst. Diese Lösung ermöglichte einen schnellen Projektstart ohne eigene Server-Infrastruktur, führte jedoch zu Abhängigkeiten von Drittanbietern. In einzelnen Situationen war die Anwendung aufgrund von Netzwerkrestriktionen, insbesondere innerhalb des Hochschulnetzes, zeitweise nicht erreichbar, da Supabase blockiert wurde. Zudem kam es vereinzelt zu Verzögerungen und Performance-Einschränkungen, die durch das Zusammenspiel von Supabase und Vercel beeinflusst wurden. Diese Erfahrungen verdeutlichten die Notwendigkeit, die Auswahl externer Dienste im Hinblick auf deren Vor- und Nachteile sowie potenzielle Risiken bewusst abzuwägen.

Auch in der Kundenkommunikation zeigte sich Verbesserungspotenzial. Technische Erklärungen bei Kritikpunkten reichten oftmals nicht aus. Stattdessen wurde deutlich, dass eine Rückmeldung über das Verständnis des Problems und die Aussicht auf eine Lösung

für den Kundenrelevanter war. Dies unterstrich die Bedeutung einer zielgerichteten Kommunikation, angepasst an die Perspektive des Kunden.

Im Bereich Dokumentation zeigte sich ebenfalls Optimierungsbedarf. Die Dokumentation wurde teilweise verspätet oder unvollständig erstellt. Die kontinuierliche Pflege der Dokumentationsinhalte erwies sich jedoch als notwendig, um den Projektverlauf nachvollziehbar zu gestalten.

Zudem fehlte eine konsequente Aufwandsschätzung während der Bearbeitung einzelner Aufgaben. Erst im späteren Projektverlauf entwickelte sich ein besseres Gespür für die Dauer einzelner Arbeitsschritte. Zukünftig wäre eine bewusste, regelmäßige Schätzung von Aufwänden sinnvoll, um realistischere Zeitpläne zu erstellen.

15.6 Fazit

Das Semesterprojekt ermöglichte den Erwerb technischer, methodischer und sozialer Kompetenzen.Diese Erfahrungen bilden eine Grundlage für zukünftige Projekte und tragen zur Weiterentwicklung im akademischen und beruflichen Kontext bei.

16 Interaktion mit Künstlicher Intelligenz

Dieses Kapitel wurde mithilfe eines KI-Modells automatisch generiert. Für die Unterstützung bei technischen Fragestellungen sowie zur Qualitätssicherung von Code und Dokumentation kam im Rahmen des Projekts das KI-Modell **ChatGPT-4o** von OpenAI zum Einsatz.

Die Künstliche Intelligenz wurde insbesondere in zwei zentralen Bereichen eingebunden:

- **Fehleranalyse im Code:** Während der Implementierungsphase wurde das Modell genutzt, um bei auftretenden Fehlern im Quellcode zu unterstützen. Durch gezielte Eingabe von Fehlermeldungen oder Codefragmenten konnten potenzielle Ursachen schneller identifiziert und behoben werden. Auch Hinweise zu Best Practices und Optimierungen wurden durch die KI eingebracht.
- **Prüfung und Optimierung der Dokumentation:** Das Modell diente außerdem zur sprachlichen und strukturellen Überarbeitung einzelner Abschnitte der technischen Dokumentation. Dabei wurden unter anderem formale Inkonsistenzen erkannt, Glossarbegriffe ergänzt und der Satzbau verbessert. Zudem wurde die Glossarstruktur auf inhaltliche Vollständigkeit geprüft.

Der Einsatz der KI erfolgte ausschließlich unterstützend. Die endgültigen Entscheidungen zu inhaltlicher Gestaltung, technischen Lösungen und Formulierungen lagen weiterhin beim Projektteam.

Hinweis: Dieser Abschnitt wurde automatisiert durch das KI-Modell ChatGPT-4o (OpenAI, 2025) erstellt.

A UI Entwürfe

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

A.1 Login

Abbildung 15: Login

A.2 Events

A.2.1 Event-Feed

Abbildung 16: Event-Feed

A.2.2 Meine Events

Abbildung 17: Meine Events

A.2.3 Event erstellen

Abbildung 18: Event erstellen

A.2.4 Umfrage erstellen

Abbildung 19: Umfrage erstellen

A.2.5 Umfrage bearbeiten

Abbildung 20: Umfrage bearbeiten

A.2.6 Meine Umfragen

Abbildung 21: Meine Umfragen

A.2.7 Event bearbeiten

Abbildung 22: Event bearbeiten

A.2.8 Event löschen

Abbildung 23: Event löschen

A.3 Wünsche

A.3.1 Wünsche-Feed

Abbildung 24: Wünsche-Feed

A.3.2 Meine Wünsche

Abbildung 25: Meine Wünsche

A.3.3 Wunsch erstellen

Abbildung 26: Wunsch erstellen

Glossar

API Abkürzung für Application Programming Interface Schnittstelle, die es ermöglicht, Funktionen und Daten eines Softwaresystems von außen programmgesteuert zu nutzen. 18

Backend Technischer Teil einer Anwendung, in dem Logik, Datenverarbeitung und Datenbanken laufen. 5

Balsamiq Software-Tool zum schnellen Erstellen von Wireframes (visuelle Entwürfe von Benutzeroberflächen). 28

CRUD Abkürzung für Create, Read, Update, Delete – Grundoperationen auf Daten in Webanwendungen. 18

Debugging Systematisches Auffinden und Beheben von Fehlern im Code. 32

Deployment Veröffentlichung einer neuen Version der Anwendung auf einem Server oder einer Plattform. 28

Docker Plattform zur Containerisierung von Anwendungen zur Sicherstellung von portabler und konsistenter Laufzeitumgebung. 6

End-to-End-Test Test der gesamten Anwendung vom Start bis zur Benutzerinteraktion, um das Zusammenspiel aller Komponenten zu prüfen. 32

Exploratory-Tests Freies Testen ohne feste Test-Skripte, um Schwachstellen oder unerwartetes Verhalten zu entdecken. 32

Feature-Branch Separater Entwicklungszweig im Versionskontrollsystem (z. B. Git), in dem neue Funktionen entwickelt werden. 32

Frontend Teil einer Anwendung, der für Nutzer sichtbar ist (Oberfläche, Benutzerinteraktion). 5

In-App-Benachrichtigungen Direkt in der Anwendung angezeigte Hinweise oder Mitteilungen für die Nutzer. 28

JWT JSON Web Token – Kompakt formatierter Token zur sicheren Übertragung von Informationen zwischen Parteien als JSON-Objekt. 18

Merge-Konflikt Konflikt beim Zusammenführen von Änderungen im Code, wenn sich Änderungen widersprechen. 32

Next.js Framework für die Entwicklung von Webanwendungen mit React, inklusive Serverfunktionen. 5

Prisma ORM-Tool (Object-Relational Mapping), das den Zugriff auf Datenbanken vereinfacht. 18

React JavaScript-Bibliothek zur Erstellung von Benutzeroberflächen, insbesondere für Single-Page-Applications. 5

Refactoring Umstrukturierung von Code, um ihn lesbarer, wartbarer oder effizienter zu machen, ohne Funktionalität zu ändern. 28

REST Architekturstil für Webservices, der auf HTTP-Protokollen basiert und CRUD-Operationen unterstützt. 18

Supabase Cloud-basierte Backend-Plattform, die unter anderem Datenbanken, Authentifizierung und APIs bereitstellt. 6

TypeScript Typsichere Programmiersprache, die auf JavaScript basiert und häufig in modernen Webanwendungen eingesetzt wird. 5

Unit-Test Test auf kleinster Ebene, bei dem einzelne Funktionen oder Methoden isoliert überprüft werden. 30

Upvote-Feature Funktion, mit der Nutzer Inhalte (z. B. Wünsche) positiv bewerten können. 28

Vercel Hosting-Plattform, die speziell auf moderne Webanwendungen wie z. B. Next.js ausgerichtet ist. 28

B Literaturverzeichnis