

# Studienarbeit

# Programmmentwurf Mobile Computing

# Kalenderapp

Fachbereich Elektrotechnik und Informationstechnik  
An der Dualen Hochschule Baden-Württemberg  
Center of Advanced Studies

**Eingereicht von:**  
Ruslan Adilgereev  
Nadine Abu El Komboz  
Lukas Renz

**Matrikelnummer:**  
7719665  
5618268  
4681850

**Jahrgang**  
2023  
2024  
2024

**Betreuer:**

Prof. Dr. Kai Becher

---

## Ehrenwörtliche Erklärung

Hiermit erkläre wir, dass die vorliegende Studienarbeit mit dem Thema

*Programmentwurf Mobile Computing*

selbstständig und ohne fremde Hilfe verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Aus fremden Quellen direkt oder indirekt übernommene Gedanken habe ich als solche kenntlich gemacht. Die Arbeit habe wir bisher keinem anderen Prüfungsamt in gleicher oder vergleichbarer Form vorgelegt. Sie wurde bisher auch nicht veröffentlicht.

Heilbronn, 6. Dezember 2024

---

Nadine Abu El Komboz

Ruslan Adilgereev

Lukas Renz

# Inhaltsverzeichnis

<b>Inhaltsverzeichnis</b>	<b>III</b>
<b>1 Einleitung</b>	<b>1</b>
<b>2 Ausführen der Software</b>	<b>2</b>
<b>3 Fachliche Beschreibung der Systemarchitektur</b>	<b>3</b>
3.1 Frontend (Flutter App) . . . . .	3
3.2 Backend (Node.js mit Express) . . . . .	3
3.3 Datenbank (MySQL) . . . . .	4
3.4 E-Mail-Service (SMTP-Server) . . . . .	4
3.5 Zusammenfassung . . . . .	4
<b>4 Umfassende fachliche Beschreibung des gezeigten Sequenzdiagramms</b>	<b>5</b>
4.1 Allgemeiner Zweck . . . . .	5
4.2 Beispiel: Authentifizierung . . . . .	5
4.3 Training Management . . . . .	6
4.4 Booking Management . . . . .	6
4.5 Termin- und Pausenplanung (Schedule Management) . . . . .	6
4.6 Besonderheiten . . . . .	7
<b>5 Umfassende Beschreibung der Datenbankstruktur</b>	<b>8</b>
5.1 Zentrale Entität: Schulungen . . . . .	8
5.2 Zeitliche Struktur: schulung_sessions, schulung_pause, schulungs- termin, schulungstermin_zeitabschnitt . . . . .	8
5.3 Tagging: tags und schulungen_tags . . . . .	9
5.4 Dozenten und Users . . . . .	9
<b>6 Herausforderungen und Lernkurve im Projektverlauf</b>	<b>14</b>
<b>7 Testing</b>	<b>15</b>
7.1 Unit-Test . . . . .	15
7.2 Widget-Test . . . . .	15
7.3 Implementierung . . . . .	15

---

<b>8</b>	<b>Bedienung durch den Benutzer</b>	<b>17</b>
8.1	Homescreen . . . . .	17
8.2	Optionen Darstellung . . . . .	17
8.2.1	Kalenderansicht . . . . .	19
8.2.2	Schulungsinformationen . . . . .	19
8.2.3	Anbieteransicht . . . . .	21
<b>9</b>	<b>Fazit und Ausblick</b>	<b>22</b>
<b>A</b>	<b>Anhang</b>	<b>23</b>
A.1	ergebnisse Testing . . . . .	23

# 1 Einleitung

In der folgenden Dokumentation wird die Implementierung einer Kalender-App beschrieben, mit der Trainingseinheiten gebucht werden können. Die Dokumentation erläutert, wie diese Anwendung konzipiert und umgesetzt wurde. Aus der Aufgabenstellung ergaben sich die folgenden Vorgaben für die Bearbeitung:

- Anbindung einer Datenbank mittels MariaDB,
- Implementierung einer Kalenderjahresansicht,
- Farbschema bestehend aus den Farben Blau, Grau und Weiß.

Zusätzlich zu diesen vorgegebenen Features konnten noch die folgenden Funktionen integriert werden:

- Startseite der App mit allgemeinen Informationen und rechtlichen Hinweisen,
- weitere Kalenderansichten (Jahr, Monat, Woche),
- Darstellung der Trainingseinheiten in einer Listenansicht (alle Trainings, gebuchte Trainings),
- Sortierfunktion für die Listenansichten,
- E-Mail-Bestätigung bei der Anmeldung,
- Implementierung von Anbieter- und Benutzerrollen:
  - Anbieter können Trainings und Trainer hinzufügen,
  - Benutzer können Trainings buchen.
- zusätzliches Farbschema im Dark Mode,
- Filterfunktionen (z. B. Suche nach Begriffen und Tags).

## 2 Ausführen der Software

Um die Software ausführen zu können, müssen sowohl Flutter als auch Dart installiert sein. Zudem müssen die in der *"pubspec.yaml"*-Datei angegebenen Pakete installiert werden. Dies erfolgt durch Ausführen des folgenden Befehls in der Konsole, während man sich im Projektordner befindet:

```
1 flutter pub get
```

### Codeausschnitt 2.1 Installieren der benötigten Pakete

Für die Einbindung der MariaDB-Datenbank sind möglicherweise Anpassungen in der Datei *"flutter\_backend/.env"* erforderlich, um den Port und den Host korrekt zu konfigurieren.

Zum Starten der App muss zunächst das Backend gestartet werden. Dies geschieht durch Ausführen des folgenden Befehls im Ordner *"flutter\_backend"*:

```
1 npm run dev
```

### Codeausschnitt 2.2 Starten des Backends

Um das Frontend zu starten, muss in einer weiteren Konsole der folgende Befehl ausgeführt werden, während man sich im Projektordner befindet. Dadurch wird das Frontend am richtigen Port gestartet:

```
1 flutter run --web-port 3000
```

### Codeausschnitt 2.3 Starten des Frontends

Um sich als Anbieter in der Software anzumelden gibt es einen Admin account, welcher *admin* sowohl als Benutzer wie auch als Passwort verwendet.

## 3 Fachliche Beschreibung der Systemarchitektur

Die vorgestellte Architektur zeigt den Aufbau einer mobilen Flutter-Applikation, die klar zwischen Frontend, Backend und Datenhaltung trennt. Ziel ist eine robuste, erweiterbare sowie wartungsfreundliche Lösung, die eine effiziente Kommunikation zwischen den Komponenten ermöglicht.

### 3.1 Frontend (Flutter App)

Die Flutter-App stellt die grafische Oberfläche bereit, nimmt Nutzereingaben entgegen und zeigt Inhalte wie Trainingsangebote, Buchungsdetails oder Suchergebnisse an. Eine vorgelagerte Service-Schicht führt REST-API-Aufrufe durch, verarbeitet eingehende Daten aus dem Backend und stellt über State Management einen konsistenten UI-Zustand sicher.

### 3.2 Backend (Node.js mit Express)

Das Backend gliedert sich in thematisch getrennte APIs:

- **Auth API:** Verantwortlich für Authentifizierung, Session-Management und Zugriffsrechte.
- **Training API:** Stellt Endpunkte zum Abruf, Erstellen und Bearbeiten von Trainingsangeboten bereit.
- **Booking API:** Ermöglicht das Reservieren, Stornieren und Verwalten von Buchungen.
- **Search API:** Unterstützt Suche und Filterung von Trainingsangeboten.

Der Express-Server dient als zentrale Kommunikationsschnittstelle, leitet Requests an die richtigen API-Routen weiter, koordiniert Datenbankzugriffe und steuert bei Bedarf weitere Dienste (z. B. E-Mail-Versand).

---

## 3.3 Datenbank (MySQL)

Eine MySQL-Datenbank, angebunden über einen Connection Pool, stellt performante, relationale Datenstrukturen bereit. Wichtige Entitäten sind:

- **Users:** Nutzerprofile und Logininformationen
- **Trainings:** Trainingsangebote mit Titeln, Beschreibungen, Terminen und Dozenten
- **Bookings:** Buchungsdaten zu einzelnen Nutzern und Kursen
- **Tags:** Zur thematischen Kategorisierung von Angeboten

## 3.4 E-Mail-Service (SMTP-Server)

Automatisierte Nachrichten wie Buchungsbestätigungen oder Passwort-Resets werden über einen SMTP-Server versendet.

## 3.5 Zusammenfassung

Das Gesamtsystem trennt UI (Flutter) von Logik (Node.js-Backend) und Datenhaltung (MySQL), integriert definierte REST-APIs und nutzt einen E-Mail-Service für Benachrichtigungen. Diese Aufteilung gewährleistet Flexibilität, Skalierbarkeit und erleichtert Wartung sowie Weiterentwicklung.



## 4 Umfassende fachliche Beschreibung des gezeigten Sequenzdiagramms

Das Sequenzdiagramm zeigt die zeitliche Abfolge konkreter Interaktionen zwischen Flutter-App, Node.js-API, MySQL-Datenbank und E-Mail-Service. Es verdeutlicht die Prozessschritte bei Authentifizierung, Trainingsverwaltung, Buchungen sowie Termin- und Pausenplanung. Statt der allgemeinen statischen Architektur veranschaulicht es, wie einzelne Funktionen operativ zusammenspielen.

### 4.1 Allgemeiner Zweck

Das Sequenzdiagramm dient dazu, den Fluss von Anfragen und Antworten in chronologischer Reihenfolge darzustellen. Vertikale Linien repräsentieren beteiligte Komponenten, horizontale Pfeile deren Kommunikation. Es zeigt, welche Daten wann angefragt, geprüft und zurückgeliefert werden. So entsteht ein tieferes Verständnis dafür, wie scheinbar einfache Nutzeraktionen komplexe interne Abläufe auslösen.

### 4.2 Beispiel: Authentifizierung

Ein Login-Request der Flutter-App an die Node.js-API führt zu einer Datenbankabfrage zur Verifizierung. Bei korrekten Credentials wird ein JWT-Token erzeugt und an die App zurückgesendet. Das Diagramm macht transparent, dass ein einzelner Login-Vorgang aus mehreren Teilschritten besteht: Eingabe prüfen, Datenbankabfrage, Token generieren, Antwort zurückgeben.

## 4.3 Training Management

Anfragen wie `GET /api/trainings` liefern strukturierte Listen verfügbarer Kurse. Hierdurch wird klar, dass die Node.js-API nicht einfach Daten durchreicht, sondern diese gezielt aus verschiedenen Tabellen aggregiert. Beim Erstellen neuer Trainings über einen POST-Request validiert die API eingehende Daten, legt neue Einträge in der Datenbank an und bestätigt den Erfolg. Ähnliches gilt für komplexere Funktionen wie die Suche nach Trainings (`GET /api/trainings/search`), wo Eingabefilter übernommen, passende Datensätze gefiltert und resultierende Listen zurückgegeben werden. Teilweise starten diese Abläufe nachgelagerte Prozesse, z. B. das Versenden von Benachrichtigungen über den E-Mail-Service, um andere Stakeholder auf neue Trainings hinzuweisen.

## 4.4 Booking Management

Beim Buchen von Terminen sind Zustandsabfragen entscheidend. Eine Buchungsanfrage (`POST /api/bookings`) prüft zunächst Verfügbarkeiten und Kontingente. Sind Kapazitäten vorhanden, wird ein neuer Datensatz eingetragen, der Erfolg bestätigt und bei Bedarf eine Bestätigungs-E-Mail versendet. Das Sequenzdiagramm verdeutlicht so die interne Logik: Datenvalidierung, Aktualisierung von Verfügbarkeiten und Rückmeldung an den Nutzer geschehen in koordinierter Reihenfolge.

## 4.5 Termin- und Pausenplanung (Schedule Management)

Funktionen zur Verwaltung von Terminen und Pausen demonstrieren, wie zeitliche Strukturen abgebildet werden. GET-Requests liefern Terminübersichten, POST-Requests legen neue Pausen fest. Auch hier durchläuft die Anfrage eine Kette von Überprüfungen und Datenbankoperationen, bevor die App entsprechende Informationen zurückerhält. Das Diagramm zeigt, wie diese technischen Schritte zusammenwirken, um letztlich eine nutzerfreundliche Terminübersicht bereitzustellen.

---

## 4.6 Besonderheiten

Das Sequenzdiagramm verdeutlicht, dass die API nicht nur reine Vermittlungsschicht ist, sondern aktiv Logik einbringt: Validierung von Eingaben, Datenverknüpfungen, Prüfung von Geschäftsregeln und Zuständen. Die Integration des E-Mail-Services macht zudem deutlich, dass externe Dienste nahtlos eingebunden sind.

Insgesamt liefert das Sequenzdiagramm einen praxisnahen Einblick in die interne Funktionsweise. Wer es betrachtet, versteht besser, wie aus simplen Nutzeraktionen – wie etwa „Login“ oder „Kurs buchen“ – komplexe Abfolgen von Datenbankabfragen, Logik und Kommunikationsschritten entstehen. So wird nachvollziehbar, wo potenzielle Fehlerquellen liegen, wie Änderungen an einer Stelle Auswirkungen auf andere Bereiche haben und welche Rolle externe Systeme spielen.

# 5 Umfassende Beschreibung der Datenbankstruktur

Das Datenmodell bildet die fachliche Grundlage des Systems, indem es Schulungsangebote, ihre zeitliche Struktur, Buchungen, Kategorisierungen (Tags), Dozenten sowie Nutzende klar abbildet. Die Datenbank verfolgt einen relationalen Ansatz, bei dem Primär- und Fremdschlüsselbeziehungen komplexe Szenarien unterstützen.

## 5.1 Zentrale Entität: Schulungen

Die Tabelle `schulungen` stellt einzelne Trainings oder Kurse dar. Neben `id` und `titel` finden sich Felder für `beschreibung`, `ort`, `gesamt_start`- und `enddatum`, `max_teilnehmer` sowie Verweise auf den verantwortlichen Dozenten. Diese Entität ist der inhaltliche Kern, um den sich weitere Tabellen gruppieren.

## 5.2 Zeitliche Struktur: `schulung_sessions`, `schulung_pause`, `schulungstermin`, `schulungstermin_zeitabschnitt`

Mehrtägige oder mehrmodulige Kurse lassen sich mit `schulung_sessions` in einzelne Zeiteinheiten aufschlüsseln. Jede Session hat Datum, Start- und Endzeiten, um den Kurs planbar und transparent zu machen. Pausen werden in `schulung_pause` festgehalten und verhindern etwaige Buchungen in den unterbrochenen Zeiträumen. Termine (`schulungstermin`) und feinere zeitliche Abschnitte (`schulungstermin_zeitabschnitt`) erlauben es, noch detailliertere Zeitpläne abzubilden. So entsteht eine fein granulare Modellierung von Kursstrukturen, Zeitfenstern und Pausen.

---

## 5.3 Tagging: tags und schulungen\_tags

Die Tag-Tabellen ermöglichen eine flexible Kategorisierung von Schulungen. Tags können etwa thematische Stichworte oder Schwierigkeitsgrade umfassen. Durch die n:m-Beziehung in `schulungen_tags` lassen sich Schulungen mit beliebig vielen Tags assoziieren, was dynamische Filterungen und Suchoptionen unterstützt.

## 5.4 Dozenten und Users

Die `dozenten`-Tabelle erfasst Lehrpersonal mit Kontaktdaten und Fachgebieten. Über `dozent_id` in `schulungen` wird eine klare Zuordnung hergestellt. Die `users`-Tabelle bildet die Endnutzenden ab – Teilnehmende, Administratoren oder weitere Rollen. Hier sind Login



Abbildung 5.1 Informationen zu den Trainings

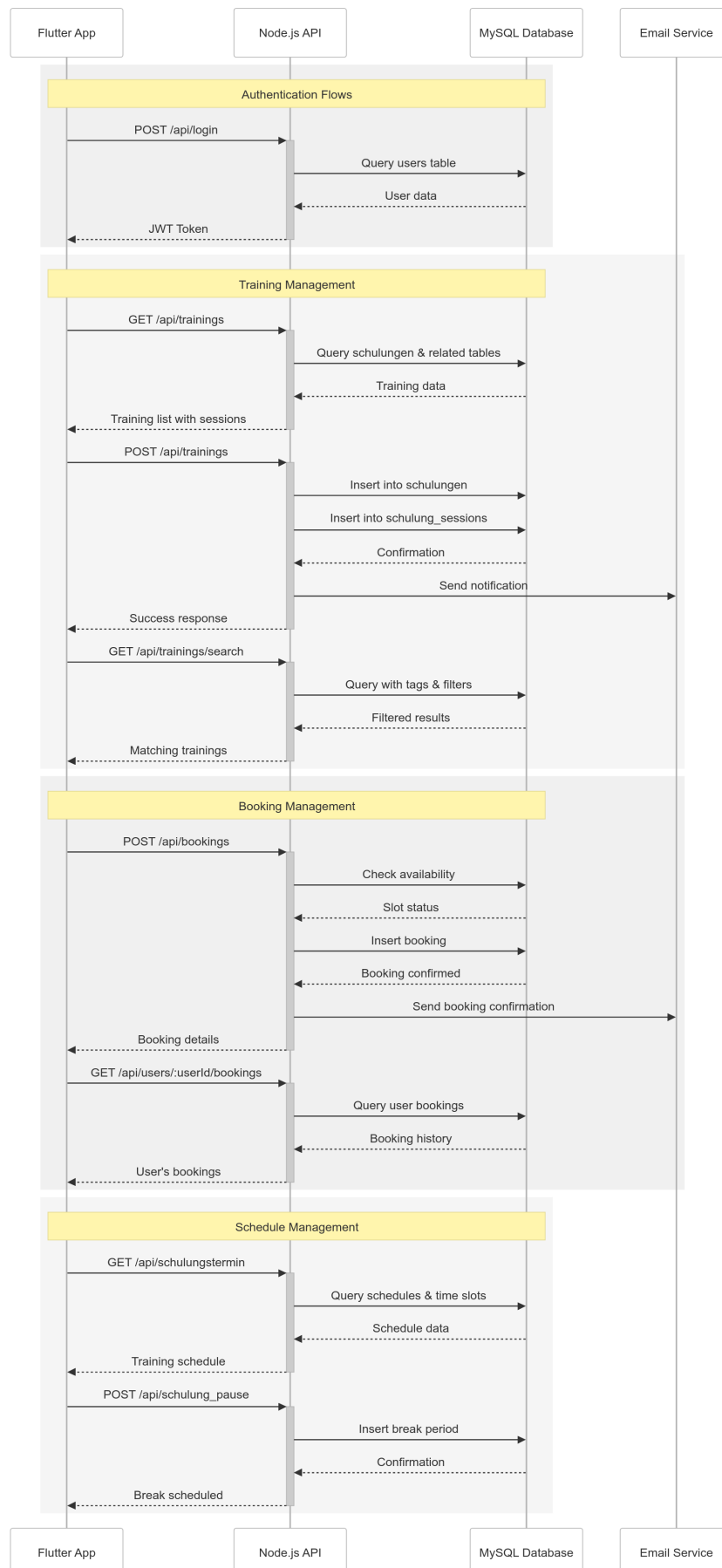


Abbildung 5.2 Ablaufsdiagramm

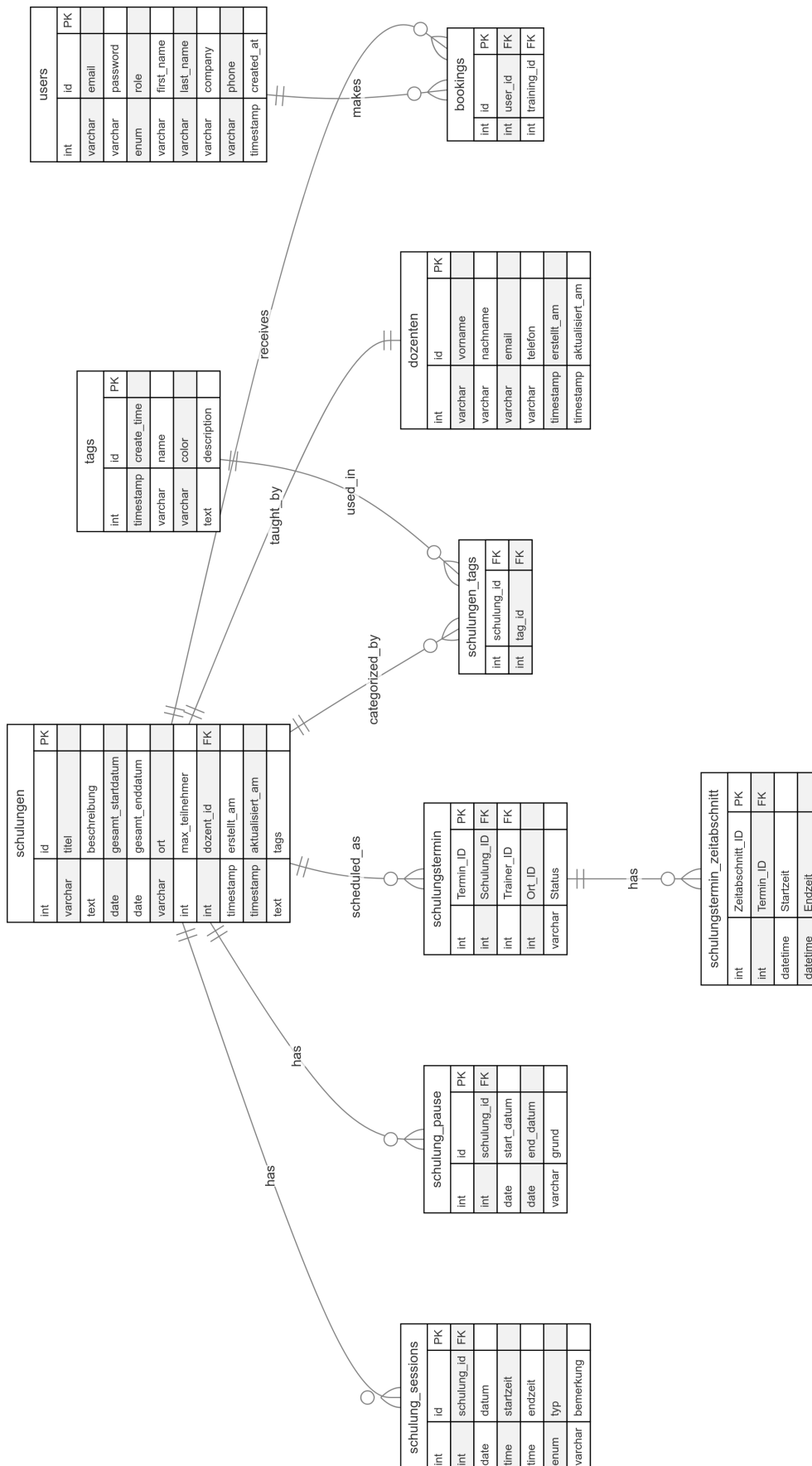


Abbildung 5.3 Datenbankstruktur



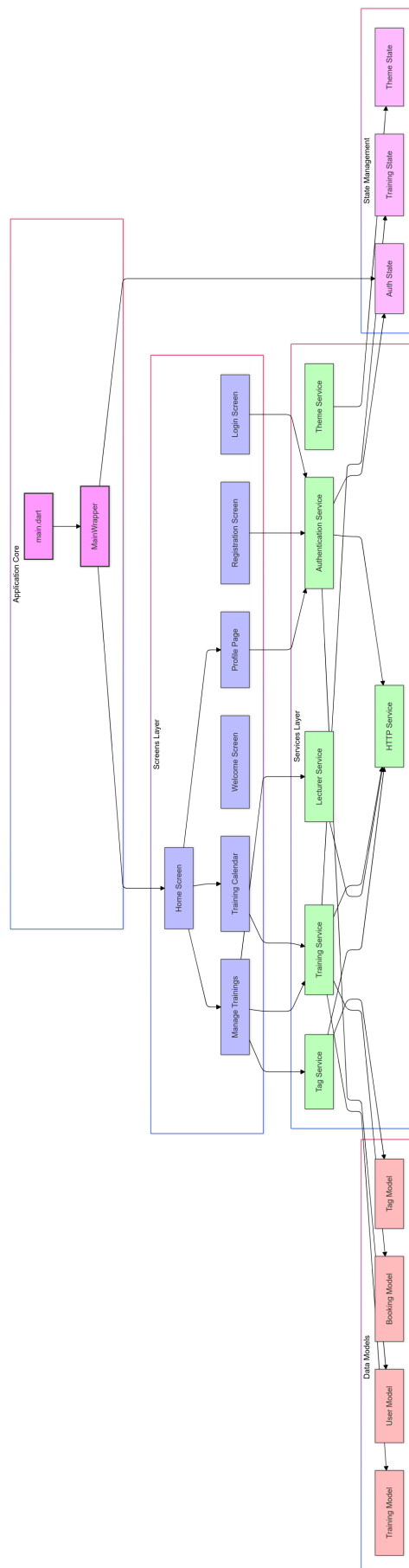


Abbildung 5.4 Component Diagramm

## 6 Herausforderungen und Lernkurve im Projektverlauf

Im Verlauf der Projektarbeit traten eine Reihe von praktischen Schwierigkeiten auf, die sowohl technische als auch konzeptionelle Aspekte betrafen. Zu Beginn gestaltete sich bereits das Einrichten der Entwicklungsumgebung und der benötigten Software, insbesondere von MariaDB, als unerwartet aufwändig. Hinzu kam, dass zunächst keinerlei Vorkenntnisse in Flutter vorlagen, weshalb eine intensive Einarbeitungsphase erforderlich war, um grundlegende Prinzipien der Flutter-Architektur, Widgets sowie State-Management zu verstehen und sicher anzuwenden.

Auch auf der UI-Ebene gab es immer wieder komplexe Herausforderungen. Das Darstellen einzelner Interface-Elemente, allen voran die sogenannte „Training Card“, führte zu unterschiedlichen Fehlerbildern in verschiedenen Ansichten der Anwendung. Parallel dazu wuchsen mit fortschreitendem Funktionsumfang die Anforderungen an die Funktionalität: Die Implementierung einer Jahresansicht für die Trainingskalender, das Hinzufügen effizienter Suchfunktionen und eine flexible Filtermöglichkeit erforderten ein tiefgreifendes Verständnis der Flutter-Widgets, asynchroner Datenabfragen sowie einer sauberen Trennung von Logik und Darstellung.

Nicht zuletzt stellte auch die Interaktion mit der Datenbank eine anspruchsvolle Aufgabe dar. Der Aufbau performanter, sicherer und gleichzeitig wartungsfreundlicher Datenbankabfragen sowie die Koppelung an die Frontend-Logik verlangten sowohl Sorgfalt als auch ein gewisses Maß an Trial-and-Error. Diese Erfahrungen führten letztlich zu einem enormen Lernzuwachs, stärkten das Verständnis für die gesamtheitliche Systemarchitektur und halfen dabei, künftige Herausforderungen systematischer und effizienter anzugehen.

# 7 Testing

Um die Kalender-App zu testen, wurden zwei Arten von Tests durchgeführt: Widget-Tests und Unit-Tests.

## 7.1 Unit-Test

Mithilfe eines Unit-Tests können einzelne Methoden und Funktionen überprüft werden. Im Test werden die erforderlichen Eingabeparameter simuliert und die Ausgabeparameter auf ihre Korrektheit hin überprüft. Ein Unit-Test greift während der Testdurchführung nicht auf die Festplatte zu. Dadurch werden weder Grafiken gerendert noch Benutzereingaben berücksichtigt.

Unit-Tests dienen daher ausschließlich dazu, die Logik einzelner Methoden zu überprüfen. Sie eignen sich jedoch nicht, um die Schnittstellenfunktionalität oder die Interaktion mit Benutzern zu testen.

## 7.2 Widget-Test

Der Widget-Test wird verwendet, um die Interaktion mit dem Benutzer und die grafische Darstellung der App zu überprüfen. In einer isolierten Testumgebung werden Widgets erzeugt und getestet, ob sie korrekt gerendert wurden. Darüber hinaus wird überprüft, ob Benutzereingaben korrekt verarbeitet werden und das erneute Rendering der Benutzeroberfläche ordnungsgemäß funktioniert.

## 7.3 Implementierung

Es wurden zwei Arten von Tests implementiert, um die Funktionalität der Kalender-App sicherzustellen:

- Unit-Tests, um die Logik einzelner Methoden innerhalb von Flutter zu überprüfen.
- Widget-Tests, um die Benutzerinteraktion und die korrekte grafische Darstellung der Anwendung zu validieren.

Der Code für die Testprogramme wurde in den folgenden Dateien umgesetzt: `"training_service_test.dart"`, `"training_service_test.mocks.dart"` und `"widget_test.dart"`. Bei `"training_service_test.dart"` und der Datei `"training_service_test.mocks.dart"` handelt es sich um den Unit-Test, wobei die Datei `"training_service_test.mocks.dart"` eine Simulationsumgebung umsetzt und die andere Datei den eigentlichen Test enthält. Durch die begrenzte Zeit wurde jeder Test jeweils für eine Methode implementiert. Mittels des Unit-Tests wurde die Funktion der Trainingsuche getestet. Dabei wurden die folgenden vier Szenarien betrachtet:

- Erfolgreiche Antworten (gültige Trainingsdaten).
- Fehlerhafte Antworten (Fehlerstatus oder ungültige JSON-Daten).
- Leere Antworten (keine Trainings gefunden).
- Formatierung von null-Werten oder fehlenden Daten.

Beim Widget-Test wurde die ordnungsgemäße Initialisierung der *ThemeService*- und *AuthService*-Provider getestet. Das Ergebnis des Testprotokolls befindet sich im Anhang.

# 8 Bedienung durch den Benutzer

## 8.1 Homescreen

Die Kalender-App wurde so konzipiert, dass sie nicht nur zur Buchung von Schulungen dient, sondern auch als Plattform genutzt werden kann, um sich als Unternehmen zu präsentieren. Die Startseite, die vor dem eigentlichen Buchungstool angezeigt wird, ist so gestaltet, dass sie allgemeine Informationen über die App bereitstellt und deren Nutzungsmöglichkeiten erläutert.

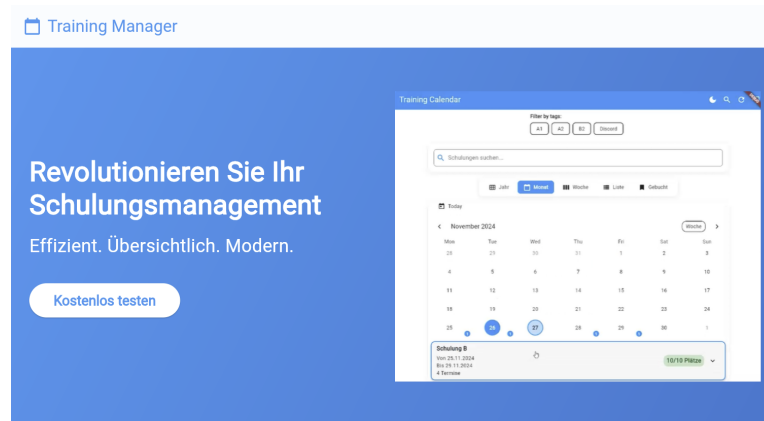
Sollte die App beispielsweise primär für Schulungen genutzt werden, könnte ein Unternehmen diese Seite dazu verwenden, sich vorzustellen. Hier könnten Informationen über das Unternehmen, seine Alleinstellungsmerkmale sowie das angebotene Leistungsspektrum bereitgestellt werden. Ebenso bietet die Startseite Platz für rechtliche Hinweise wie Datenschutzinformationen und ein Impressum, für das am unteren Seitenrand ein entsprechender Link eingebaut ist.

Von dieser Startseite gelangt der Benutzer über den Button *"Kostenlos testen"*, der in Abbildung 8.1 dargestellt ist, zur eigentlichen Kalenderübersicht. Diese bildet den zentralen Bestandteil des Buchungssystems.

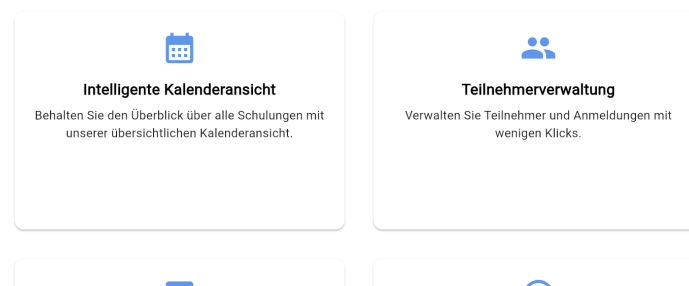
## 8.2 Optionen Darstellung

Im Kalender wurden mehrere Optionen implementiert, die in der Abbildung 8.2 dargestellt sind. Eine dieser Optionen ist die blaue Leiste, die der Navigation zwischen den einzelnen Fenstern dient und Änderungen des Farbschemas ermöglicht. Über diese Funktion gelangt man zurück zum Homescreen, öffnet ein Fenster zur Suche, wechselt das Farbschema in den Dark Mode oder kann die Einstellungen des persönlichen Profils bearbeiten. Sollte der Nutzer noch nicht angemeldet sein, wird diese Leiste als Login-Fenster angezeigt.

Unterhalb der Navigationsleiste befindet sich eine Funktion, mit der Schulungen nach bestimmten Tags gefiltert werden können. In diesem Fall wurden als



Alles was Sie brauchen

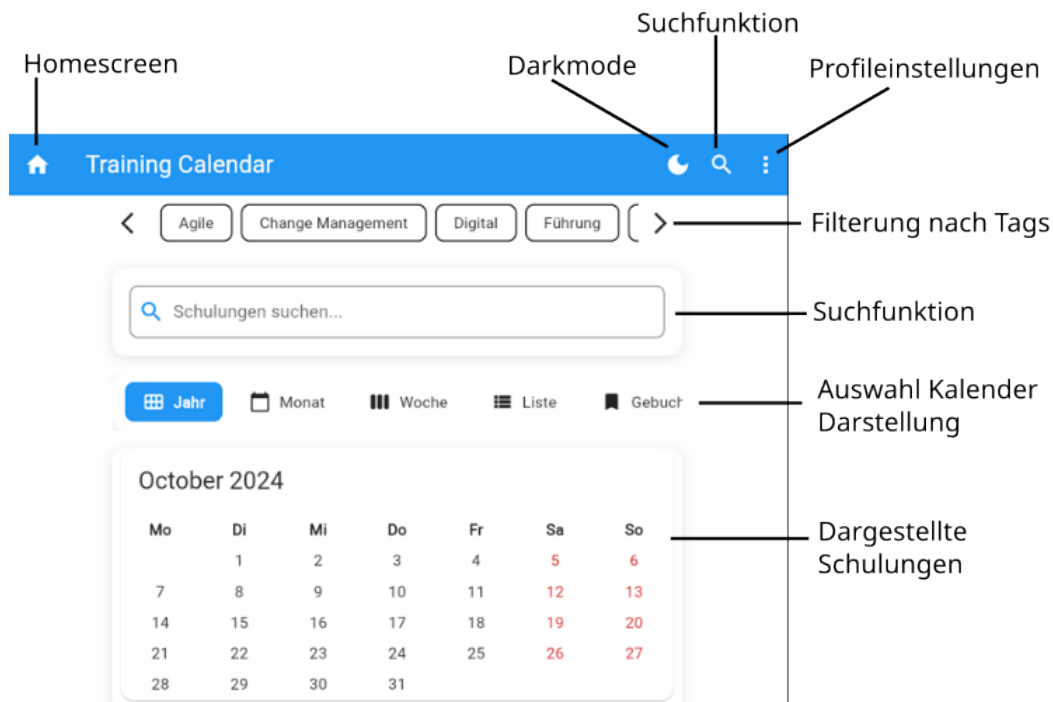


**Abbildung 8.1** Startseite

Beispiele die Fachübergreifenden Kompetenzen des CAS gewählt. Bei anderen Schulungen, wie etwa Sicherheitsunterweisungen, könnte hier auch die dazugehörige ID angezeigt werden. Durch das Anklicken eines Tags wird eine Filterung nach diesem Kriterium vorgenommen.

Unter der Tag-Filterung ist eine Freitextsuche implementiert, die es ermöglicht, durch alle relevanten Informationen eines Trainings zu suchen. So kann nach dem Trainingsnamen, dem Startdatum oder auch dem Namen des Trainers gesucht werden. Sowohl die Filterung durch die Suche als auch die Tag-Filterung wird bei der Darstellung jeder Kalenderansicht berücksichtigt.

Eine weitere Option ermöglicht es, die Schulungen in verschiedenen Zeiträumen anzuzeigen. Der Nutzer kann zwischen den Ansichten „Jahr“, „Monat“ und „Woche“ wählen, wobei die Monatsansicht die Standarddarstellung darstellt. Neben der Darstellung im Kalender kann die Schulungsliste auch in Listenform angezeigt werden. In dieser Ansicht kann der Nutzer zwischen einer Filterung nach allen Schulungen oder nur nach den gebuchten Schulungen wählen.



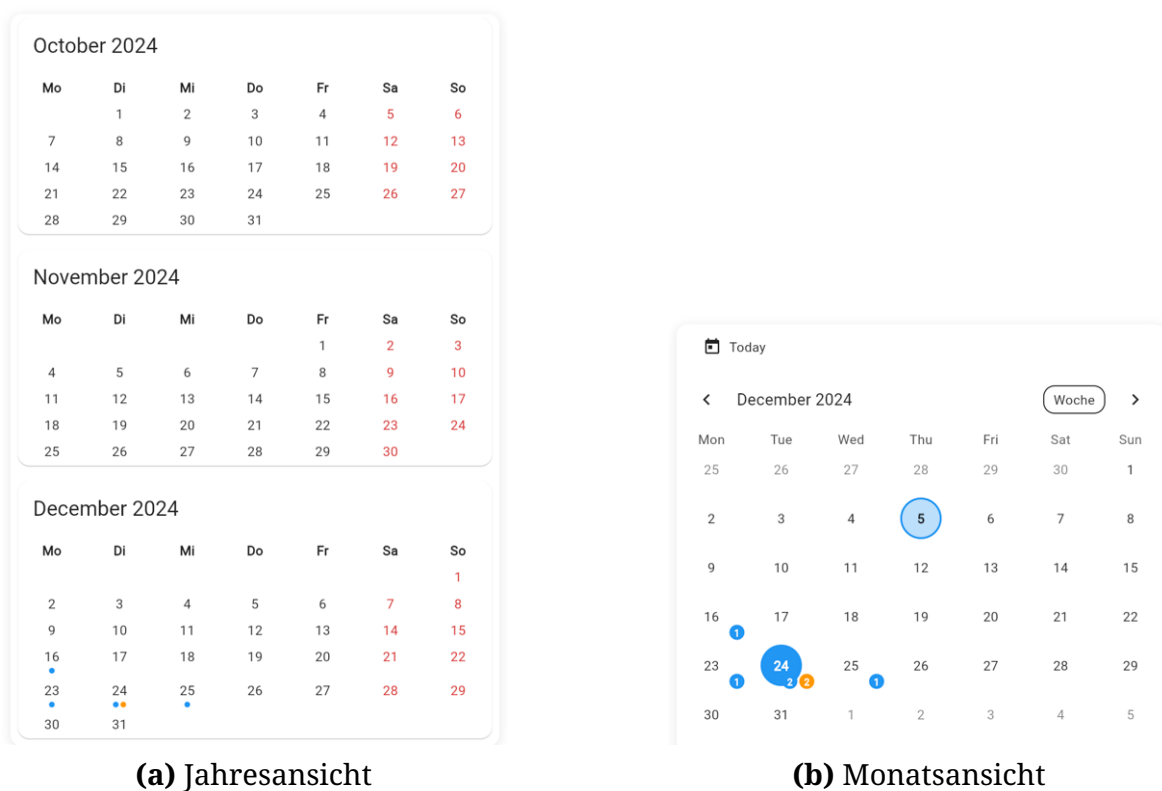
**Abbildung 8.2** Option in der Kalenderansicht

### 8.2.1 Kalenderansicht

In der Kalenderansicht werden Tage, an denen Trainings angeboten werden, mit orangen und blauen Punkten markiert. Ein orangefarbener Punkt bedeutet, dass es sich um eine mehrtägige Schulung handelt, während ein blauer Punkt darauf hinweist, dass die Schulung nur an einem Tag stattfindet. In der Monats- und Wochenansicht sind zusätzlich Nummern in diesen Punkten enthalten, die angeben, wie viele Schulungen an diesem Tag angeboten werden. Dieses unterschiedliche Verhalten ist in der Abbildung 8.3 dargestellt. Es ist ersichtlich, dass in der Jahresansicht am 24. Dezember nur zwei Punkte angezeigt werden, während in der Monatsansicht zusätzlich Zahlen zu sehen sind, die die Anzahl der Schulungen an diesem Tag angeben. Ist eine Schulung an einem Tag gebucht, wird dieser Tag zusätzlich rot umrandet.

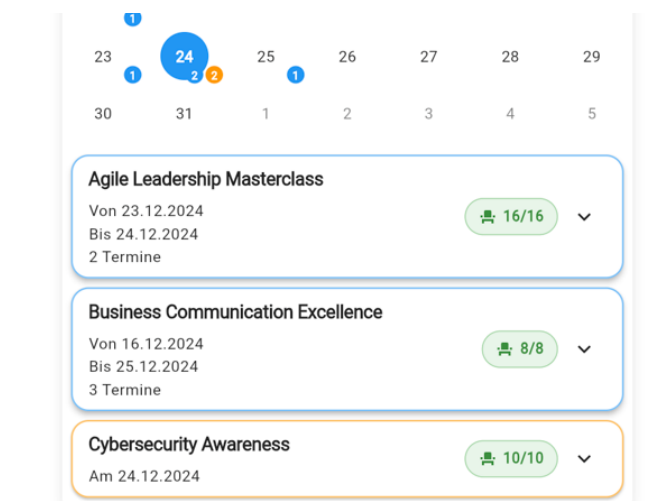
### 8.2.2 Schulungsinformationen

Um genauere Informationen zu erhalten, muss der Tag im Kalender angeklickt werden. Anschließend erscheint die Ansicht, die in Abbildung 8.4 dargestellt ist. Durch Scrollen kann man alle Schulungen durchsehen. Zudem erhält man



**Abbildung 8.3** Darstellung der unterschiedlichen Markierungen der verfügbaren Schulungen

durch das Anklicken einer Schulung weitere Informationen, wie beispielsweise die Kontaktdaten des Dozenten, eine Beschreibung der Schulung sowie die Startzeiten.



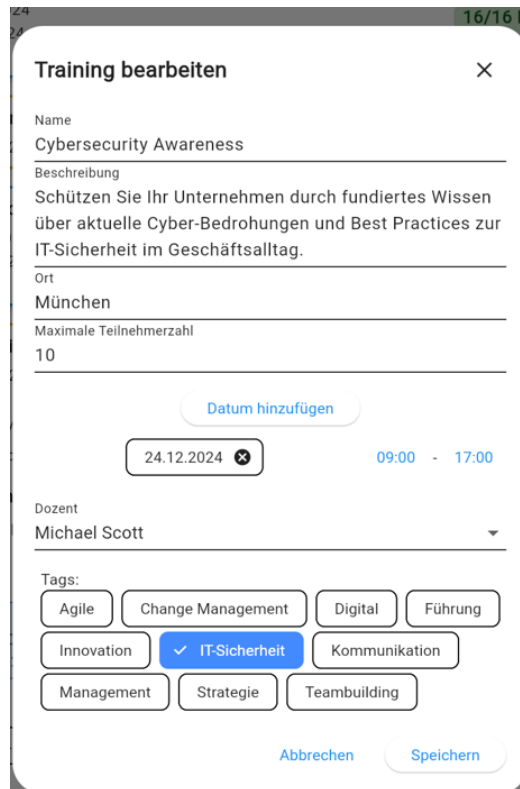
**Abbildung 8.4** Informationen zu den Trainings



Entscheidet man sich, ein Training zu buchen, so kann man einen Haken setzen, um eine Buchungsbestätigung zu erhalten. Dadurch wird dem Benutzer eine E-Mail als Buchungsbestätigung zugesendet.

### 8.2.3 Anbieteransicht

Meldet man sich als Anbieter von Schulungen in der App an, so hat man die Möglichkeit, Trainings zu erstellen und diese zu bearbeiten. In Abbildung 8.5 sind die verschiedenen Beschreibungsmöglichkeiten für eine Schulung dargestellt. Die Auswahl des Datums kann dabei in einer Kalenderansicht erfolgen, in der der gesamte Zeitraum markiert wird, in dem die Schulung stattfindet.



**Training bearbeiten** ✕

Name  
Cybersecurity Awareness

Beschreibung  
Schützen Sie Ihr Unternehmen durch fundiertes Wissen über aktuelle Cyber-Bedrohungen und Best Practices zur IT-Sicherheit im Geschäftsalltag.

Ort  
München

Maximale Teilnehmerzahl  
10

Datum hinzufügen

24.12.2024 ✕ 09:00 - 17:00

Dozent  
Michael Scott ▾

Tags:

Agile Change Management Digital Führung

Innovation **✓ IT-Sicherheit** Kommunikation

Management Strategie Teambuilding

Abbrechen Speichern

**Abbildung 8.5** Bearbeiteransicht

## 9 Fazit und Ausblick

Die vorgestellte Anwendung verdeutlicht ein in sich stimmiges Gesamtkonzept, das auf professionelle Weiterbildung und Kursverwaltung ausgerichtet ist. Im Zentrum steht eine mobile Flutter-App, die Nutzerinnen ein ansprechendes, intuitives Interface für die Buchung und Verwaltung von Schulungs- und Trainingsangeboten bietet. Die klare Trennung zwischen Frontend, Backend und Datenhaltung schafft dabei eine robuste Grundlage, um reibungslose Abläufe und hohe Performanz sicherzustellen. Nutzerinnen können Angebote durchsuchen, gezielt filtern, Termine buchen oder stornieren und erhalten durch die Integration eines E-Mail-Services automatisierte Benachrichtigungen, etwa zu bevorstehenden Kursen oder erfolgreichen Anmeldungen. Die Architektur ist so gestaltet, dass sie nicht nur die unmittelbaren Anforderungen erfüllt, sondern auch langfristige Weiterentwicklung ermöglicht. Dank sauber definierter Schnittstellen und eines klaren State-Managements lassen sich neue Funktionen, zusätzliche Kurskategorien oder weiterführende Analysewerkzeuge ohne grundsätzliche Umbrüche integrieren. Das Backend sorgt über transparente APIs und effektives Session-Management für Datensicherheit und Verlässlichkeit. Die relationale MySQL-Datenbank ermöglicht es, selbst komplexe Kursstrukturen, Terminpläne, Pausenregelungen und Nutzerprofile zentral abzubilden, was eine einfache Auswertung, Erweiterung und Pflege der Inhalte gestattet. Insgesamt zeigt sich, dass die Applikation nicht nur eine aktuelle Lösung für ein bestimmtes Anwendungsfeld darstellt, sondern ein solides Fundament für künftige Anforderungen liefert. Die modulare und gut dokumentierte Architektur trägt maßgeblich zur Wartungsfreundlichkeit, Skalierbarkeit und Flexibilität bei. Damit ist die Anwendung ideal darauf vorbereitet, mit den Erwartungen ihrer Nutzer\*innen mitzuwachsen und langfristig ein verlässlicher Begleiter im Schulungs- und Weiterbildungsbereich zu sein.

---

# A Anhang

## A.1 ergebnisse Testing

## Flutter Application Test Report

Date: 2024-01-18

### Overview

This report documents the test results for the Training Calendar Flutter application. The tests cover both unit testing of the TrainingService and widget testing of the main application.

### Test Environment

- Framework: Flutter Test
- Testing Libraries:
  - flutter\_test
  - mockito
  - build\_runner

### Unit Tests Results

#### TrainingService Tests

Location: test/training\_service\_test.dart

##### 1. Search Functionality Test ¶

**Description:** Tests the normal search functionality for trainings

**Status:** PASSED

**Verified:**

- Correct API endpoint called
- Proper data transformation
- Expected response structure

##### 2. Error Handling Test ¶

**Description:** Tests error handling for API failures

**Status:** PASSED

**Verified:**

- Exception thrown on 500 status code
- Proper error message propagation

##### 3. Empty Response Test ¶

**Description:** Tests handling of empty search results

**Status:** PASSED

**Verified:**

- Empty list returned
- No exceptions thrown

##### 4. Malformed Response Test ¶

**Description:** Tests handling of invalid JSON responses

**Status:** PASSED

**Verified:**

- Exception thrown on invalid JSON
- Proper error handling

##### 5. Data Formatting Test ¶

**Description:** Tests null value handling and data formatting

**Status:** PASSED

**Verified:**

- Default values applied for null fields
- Proper data transformation

### Widget Tests

#### App Initialization Test ¶

Location: test/widget\_test.dart