DHBW CAS
Duale Hochschule
Baden-Württemberg
Center for Advanced Studies

Studienarbeit

# Programmentwurf Mobile Computing

## Kalenderapp

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



| **Eingereicht von:** | **Matrikelnummer:** | **Jahrgang** |
|---|---|---|
| Ruslan Adilgereev | 7719665 | 2023 |
| Nadine Abu El Komboz | 5618268 | 2024 |
| Lukas Renz | 4681850 | 2024 |

| | |
|---|---|
| **Betreuer:** | Prof. Dr. Kai Becher |



Datum: 5. Dezember 2024

# 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, 5. Dezember 2024

Nadine Abu El Komboz Ruslan Adilgereev Lukas Renz

# Inhaltsverzeichnis

# 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

# 3 Bug und Revision

# 4 Testing

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

## 4.1 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.

## 4.2 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.

## 4.3 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 Trainingssuche 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.

# 5 Bedinung des Benutzers

Die Kalenderapp bietet dem Benutzer eine übersichtliche und benutzerfreundliche Bedienoberfläche, die es ermöglicht, Schulungsdaten effektiv zu verwalten und zu organisieren. Wenn man die App öffnet befindet man sich auf der Startseite 5.1. Hier befinden sich allgemeine Informationen über die App und weiter unten befindet sich ein Link zu Rechtlichem/ zum Impressum 5.2.

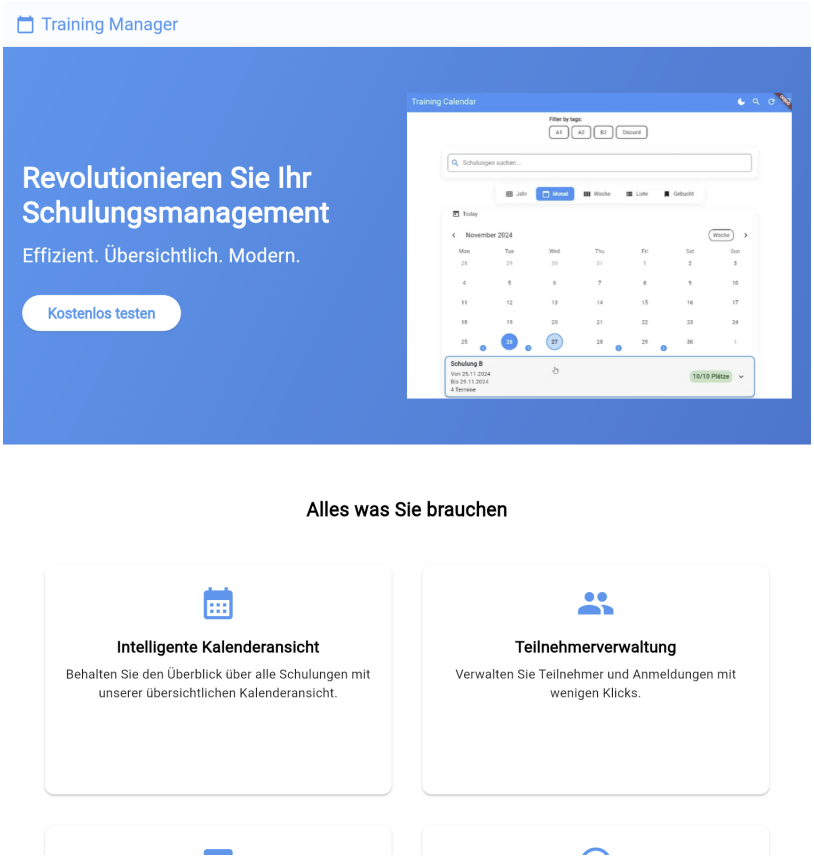



**Abbildung 5.1** Startseite

Mit dem Button “Kostenlos testen” gelangt man zum tatsächlichen Kalender. Als Benutzer kann man oben in der Navigationsleiste zwischen verschiedenen Modi wechseln, wie Jahr, Monat, Woche, Liste und einer Buchungsübersicht, indem man auf die entsprechenden Buttons klickt. Standardmäßig gelangt man zur Monatsansicht. Die App bietet eine intuitive Navigation, bei der der Benutzer zwischen den verschiedenen Ansichten hin und her wechseln kann. Alle Daten sind in einer klaren, strukturierten Form präsentiert, um eine effiziente Nutzung

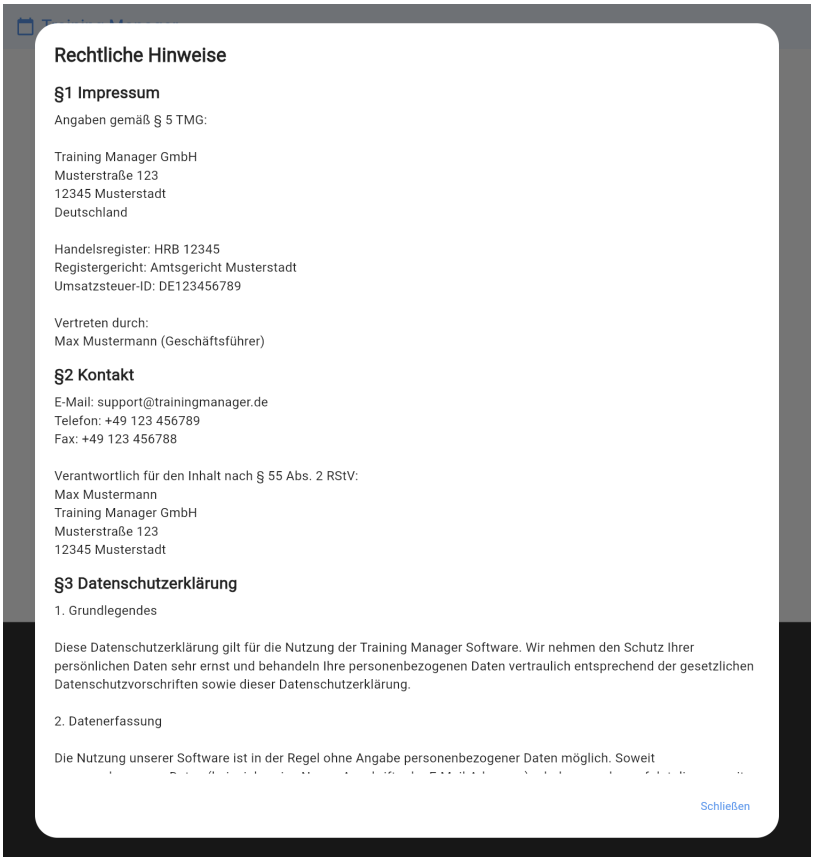



**Abbildung 5.2** Impressum

und eine schnelle Orientierung zu gewährleisten. Details zu den verschiedenen Ansichten:

- **Jahresansicht:** In dieser Ansicht (5.3) sieht der Benutzer das gesamte Jahr auf einen Blick. Alle Schulungsdaten sind übersichtlich auf dem Jahreskalender verzeichnet, sodass der Benutzer schnell einen Überblick über alle geplanten Schulungen und Termine erhält. Diese Ansicht eignet sich besonders, um langfristige Planungen und Übersichten zu erstellen.

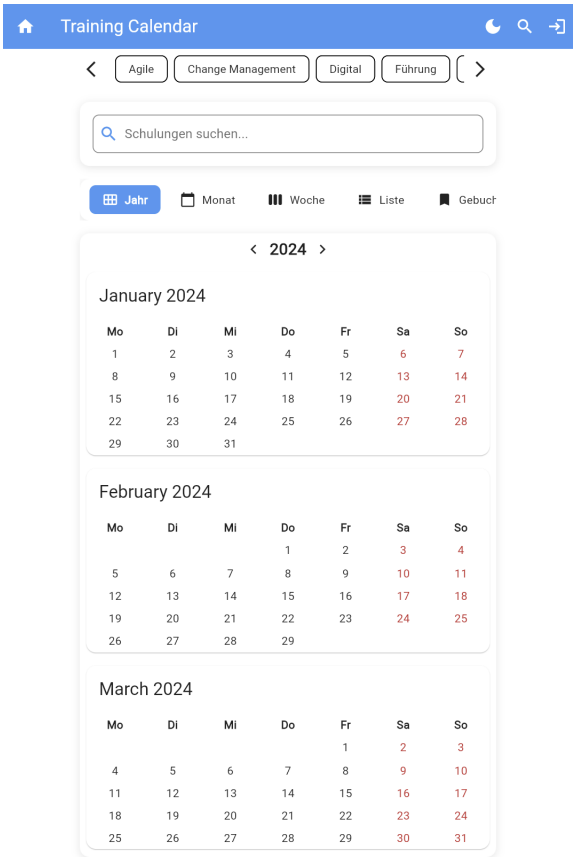



**Abbildung 5.3** Jahresansicht

- **Monatsansicht:** Hier wird der aktuelle Monat detailliert dargestellt (5.4). Die Monatsansicht zeigt alle Tage eines Monats in einer Rasterdarstellung an, wobei der aktuelle Tag hervorgehoben wird, wie beispielsweise der ——— im Screenshot. Mithilfe der Pfeiltasten links und rechts kann man zwischen den Monaten wechseln, um Termine in anderen Zeiträumen zu betrachten. Der Benutzer kann jeden einzelnen Tag im Monat aufrufen, um mehr Informationen zu den jeweiligen Schulungen zu erhalten.
- **Wochenansicht:** In dieser Ansicht (5.5) werden die Schulungsdaten in einer Wochenübersicht angezeigt. Dies ist besonders hilfreich, um kurzfristige

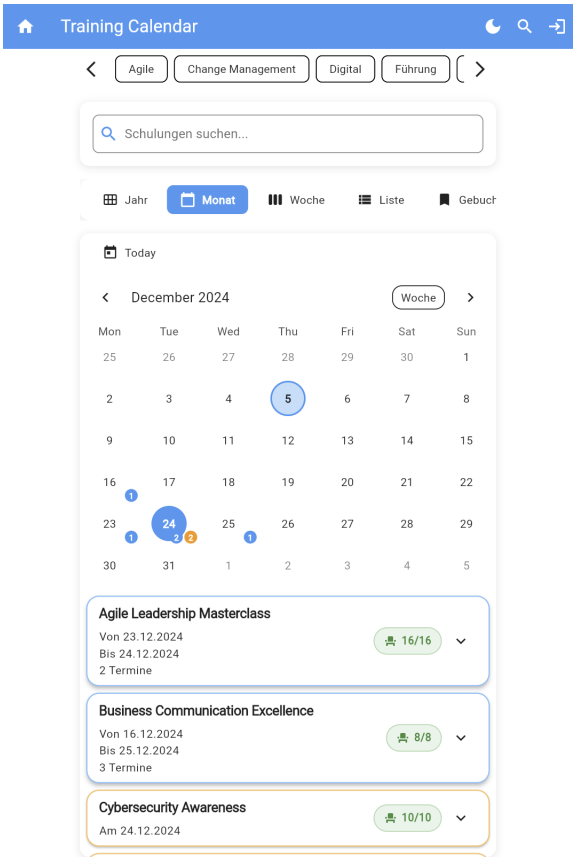



**Abbildung 5.4** Monatsansicht

Schulungen und Termine in der kommenden Woche zu überprüfen und zu verwalten. Der Benutzer hat die Möglichkeit, durch die Wochentage zu blättern und gezielt nach freien Zeiträumen zu suchen.

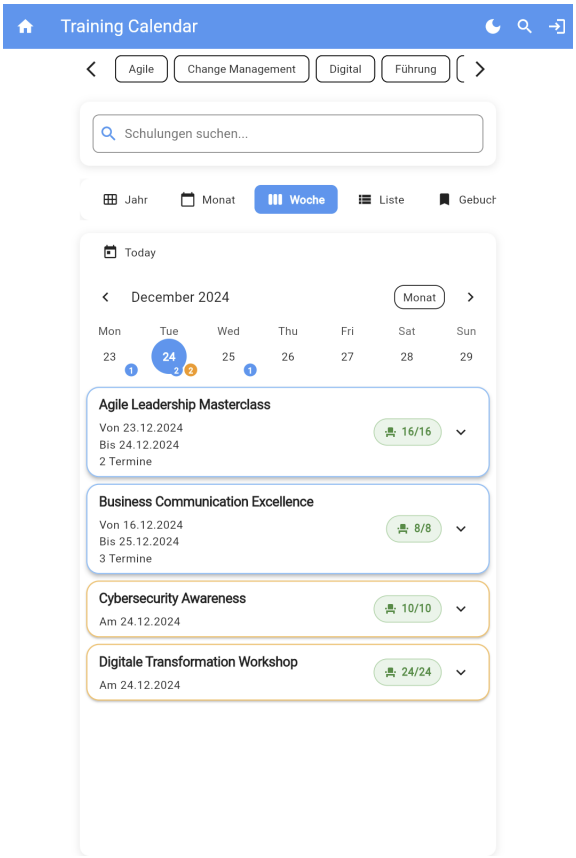



**Abbildung 5.5** Wochenansicht

- **Listenansicht:** Alle verfügbaren Schulungen sind in einer übersichtlichen Liste aufgeführt (5.6). Der Benutzer kann durch diese Liste scrollen, um Informationen zu verschiedenen Schulungen wie Titel, Datum, Uhrzeit und Ort zu erhalten. In dieser Ansicht werden auch Sortierfunktionen angeboten, mit denen der Benutzer gezielt nach bestimmten Schulungen suchen kann. Die „Liste“-Ansicht ist besonders hilfreich, um alle Termine in einer scrollbaren Liste darzustellen, was einen schnellen Überblick über anstehende Events ermöglicht.
- **Gebucht-Ansicht:** In der Gebucht-Ansicht (5.7) werden alle bereits gebuchten Schulungen des Benutzers aufgelistet. Hier kann der Benutzer seine kommenden Termine einsehen, Änderungen vornehmen oder auch Schulungen absagen, falls erforderlich. Diese Ansicht hilft dabei, den Überblick über die eigenen gebuchten Veranstaltungen zu behalten. Hier wird einem

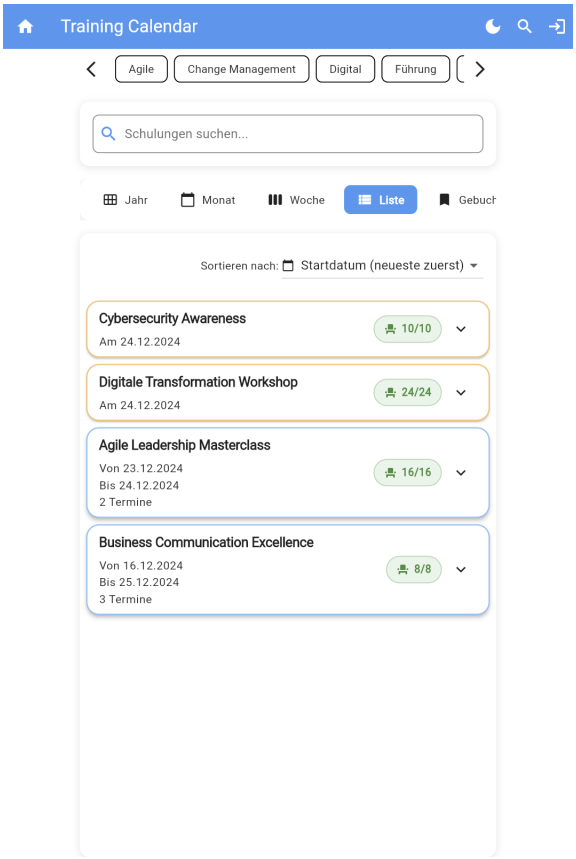



**Abbildung 5.6** Listenansicht

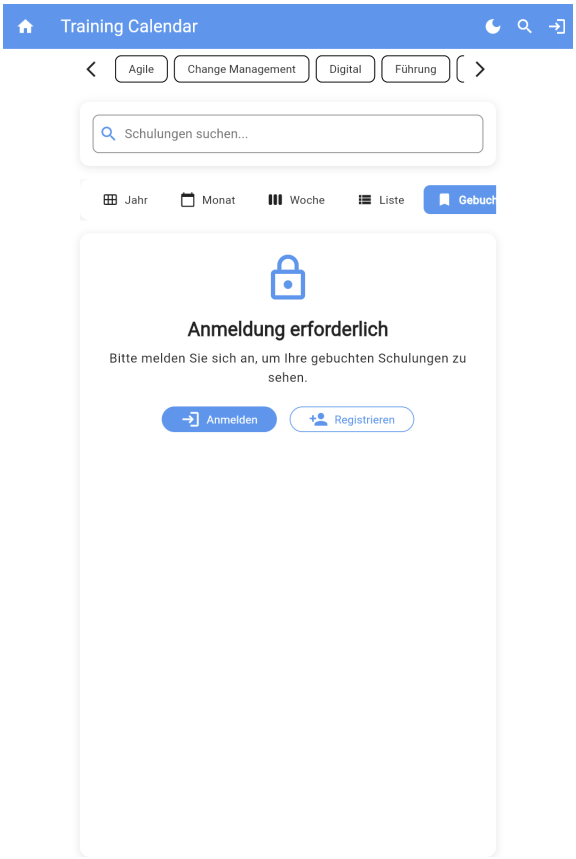



**Abbildung 5.7** Gebucht-Ansicht unangemeldet

unangemeldeten Benutzer angezeigt, dass er sich zuerst anmelden muss. So sieht die Ansicht für angemeldete Benutzer aus (5.8): Die Buchungen stehen

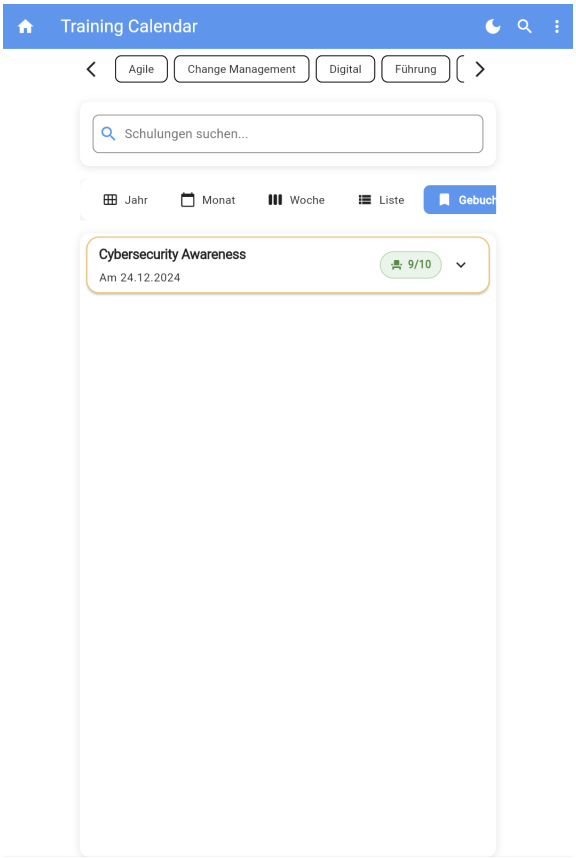



**Abbildung 5.8** Gebucht-Ansicht angemeldet

übersichtlich untereinander aufgelistet. Für eine detailreichere Ansicht kann man Buchungen mit einem Klick aufklappen (5.9): Der „Gebucht“-Button dient dazu, nur Termine anzuzeigen, die der Benutzer bereits gebucht hat.

- **Such-Ansicht:** Weiterhin gibt es eine Suchleiste ganz oben über den Ansichten. Es kann nach einem Termin, einem Schulungsnamen oder einem Dozenten gesucht werden. Oben befindet sich ein Suchfeld, in das Benutzer Stichworte oder Tags eingeben können, um gezielt nach bestimmten Schulungen oder Ereignissen zu suchen. Direkt daneben gibt es die Option, Filter nach Tags anzuwenden, um die Ansicht weiter zu personalisieren.

Durch die klare Markierung des aktuellen Tages können Benutzer schnell erkennen, welcher Tag heute ist, ohne die Ansicht manuell zu suchen. Zusätzlich kann der Benutzer mit der Nachtmodus-Schaltfläche (oben rechts) die Ansicht zwischen Tag- und Nachtmodus wechseln, was die App auch in dunklen Umgebungen angenehm lesbar macht.

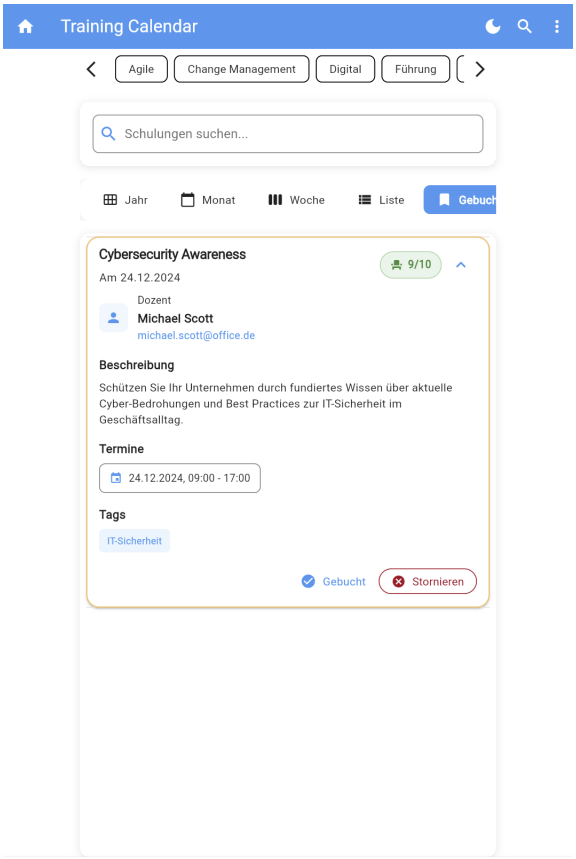



**Abbildung 5.9** Gebucht-Ansicht angemeldet ausgeklappt

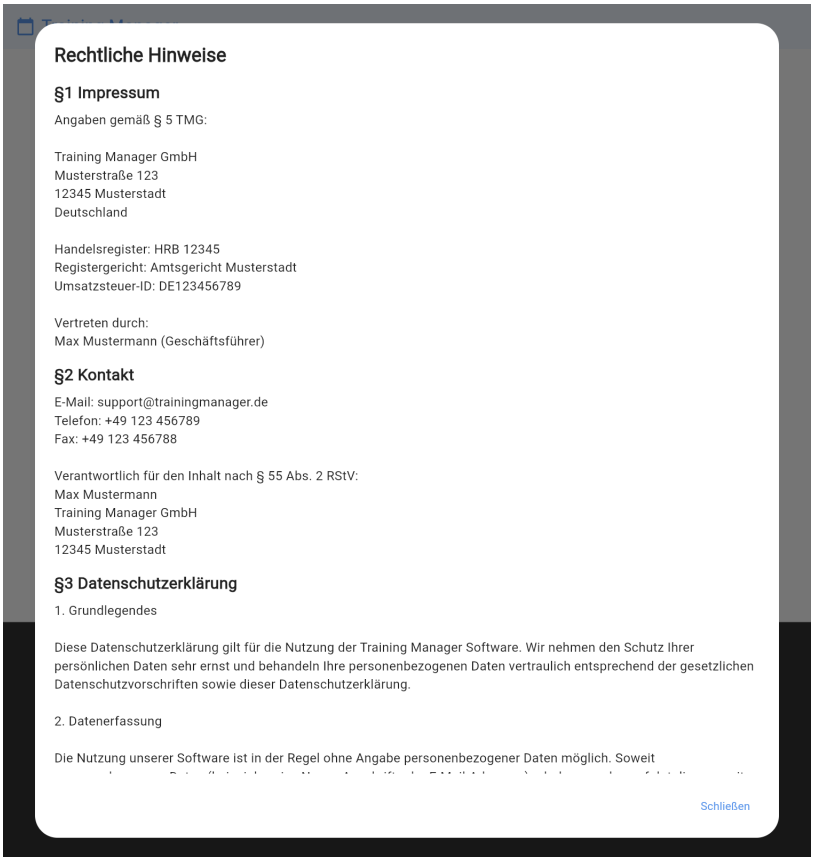
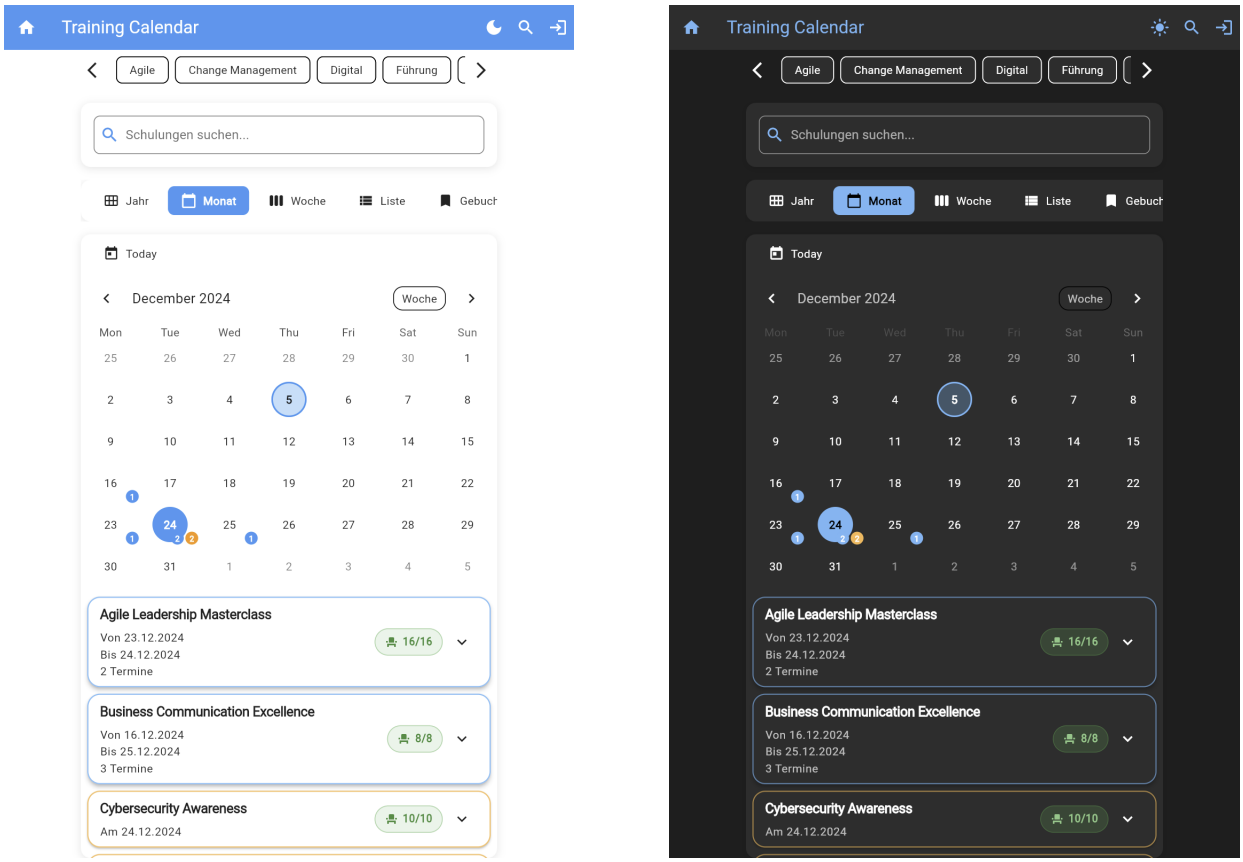



**Abbildung 5.10** Such-Ansicht




**(a)** Normale Ansicht **(b)** Ansicht im Dark mode

**Abbildung 5.11** Ansicht im Dark mode

Eine kleine Home-Schaltfläche in der oberen linken Ecke führt jederzeit zurück zur Startseite. Die App ist darauf ausgelegt, schnell und effizient zwischen verschiedenen Ansichten zu wechseln und die gesuchten Informationen mit wenigen Klicks zugänglich zu machen. Die Markierungen der Schaltflächen und das Design der Kalenderansicht sind minimalistisch und lenken nicht vom Wesentlichen ab. Durch die Kombination von Suchfunktionen und personalisierbaren Ansichten wird die Nutzung besonders benutzerfreundlich gestaltet.

# 6 Fazit und Ausblick

# 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`