ipso!

Meeting Room Reservation Projektdokumentation

Von Even, Yannick und Tunahan

Inhalt

1. Projektaufgabe

1.1 Lernziele

- Erstellung eines Projektplans
- Nutzung der Technologien: Angular/React, ASP.NET Core WebAPI und MongoDB
- Sammeln von Erfahrungen in der Teamarbeit

1.2 Aufgabenbeschreibung

Die MeetingPoint AG-Service möchte eine datenbankbasierte Web-Lösung für die Verwaltung und Reservierung interner Besprechungsräume implementieren.

2. Zielsetzung

Ziel dieses Projekts ist die Entwicklung einer einfachen Webanwendung, die es Nutzern ermöglicht, Besprechungsräume zu reservieren. Die Anwendung besteht aus:

- **Frontend:** Angular/React
- **Backend:** ASP.NET Core WebAPI
- **Datenbank:** MongoDB
- **Funktionalität:** CRUD-Operationen für Besprechungsräume und Reservierungen

3. Projektanforderungen

3.1. Frontend (Angular/React)

Das Frontend kommuniziert mit der Backend-API und bietet eine Benutzeroberfläche zur Verwaltung von Besprechungsräumen und Reservierungen.

Funktionen:

- Anzeige verfügbarer Besprechungsräume und aktueller Reservierungen
- CRUD-Operationen für Besprechungsräume (Erstellen, Lesen, Aktualisieren, Löschen)
- Listen- oder Kalenderansicht für Reservierungen
- Validierung von Eingaben (z. B. Verhinderung von Überschneidungen bei Reservierungen)
- Responsive UI und interaktive Elemente

3.2. Backend (ASP.NET Core WebAPI)

Entwicklung einer RESTful API zur Verwaltung von Besprechungsräumen und Reservierungen.

Funktionen:

- CRUD-Endpunkte für Besprechungsräume
- CRUD-Endpunkte für Reservierungen
- Datenvalidierung (z. B. Vermeidung von Terminüberschneidungen)
- Speicherung und Abruf von Daten aus MongoDB
- Logging von Laufzeitfehlern und API-Anfragen

3.3. Datenbank (MongoDB)

Collections:

- **MeetingRooms:** Enthält Raumname, Kapazität, Ausstattung und Verfügbarkeit
- **Reservations:** Enthält Start- und Endzeit, Raum-ID, Benutzer-ID und Zweck der Reservierung

Funktionen:

- Sicherstellen korrekter CRUD-Operationen
- Verhinderung von Überschneidungen bei Reservierungen
- Sicherstellen, dass nur verfügbare Räume gebucht werden können

4. Projektstruktur

Das Projekt wird in spezifische Bereiche aufgeteilt, wobei Teammitglieder jeweils bestimmte Verantwortlichkeiten übernehmen:

- **Frontend:** Entwicklung mit Angular/React
- **Backend:** Entwicklung mit ASP.NET Core WebAPI
- **Datenbank:** Design und Integration von MongoDB
- **Testing & Qualitätssicherung**

4.1 Yannick

- **Frontend (Angular/React)**:
    - Entwicklung der Benutzeroberfläche.
    - Implementierung von Formularen für Räume und Reservierungen.
    - Integration der API und Validierung von Benutzereingaben.

4.2. Even

- **Backend (ASP.NET Core WebAPI)**:
    - Entwicklung der API zum Verwalten der Besprechungsräume und Reservierungen.
    - Implementierung von CRUD-Endpunkten und Datenvalidierung.
    - Anbindung an die MongoDB-Datenbank.

4.3. Tunahan

- **Datenbank (MongoDB)**:
    - Design der Datenbank und Modelle für Besprechungsräume und Reservierungen.
    - Sicherstellung, dass nur verfügbare Räume gebucht werden können und keine Überschneidungen entstehen.

4.4 Gemeinsame Aufgaben

- **Testing & Qualitätssicherung**:
    - Alle testen gemeinsam die Anwendung. Even testet das Frontend, Yannick das Backend und Tunahan die Datenbank.
    - Durchführung von Unit-Tests und Fehlerbehebung.

4.5 Ordner Struktur

Frontend (React)

```
Frontend/
├── public/                        # Statische Dateien wie Logos
│   ├── MeetingPoint-Logo.png
│   └── ...
├── src/
│   ├── api/                       # API-Requests zu Backend
│   │   ├── apiConfig.js
│   │   ├── meetingRooms.js
│   │   └── reservations.js
│   ├── components/                # Wiederverwendbare UI-Komponenten
│   │   ├── ReservationCalendar.jsx
│   │   ├── ReservationForm.jsx
│   │   ├── RoomCard.jsx
│   │   └── RoomForm.jsx
│   ├── layouts/                   # Layout-Komponenten wie Navigation
│   │   └── MainLayout.jsx
│   ├── pages/                     # Seiten: Dashboard, Räume, Reservierungen
│   │   ├── Dashboard.jsx
│   │   ├── Reservations.jsx
│   │   └── Rooms.jsx
│   ├── styles/
│   │   └── global.css             # Globale Styles & Kalenderanpassungen
│   ├── utils/
│   │   └── validation.js          # Validierungslogik für Reservierungen
│   ├── App.jsx
│   ├── App.css
│   ├── main.jsx
│   └── theme.js                   # MUI-Theme-Konfiguration
├── eslint.config.js               # ESLint Setup für Codequalität
├── index.html
├── vite.config.js
└── package.json
```

Backend (ASP.NET Core WebAPI)

```
MeetingRoomReservationAPI/
├── Controllers/
│   ├── MeetingRoomsController.cs
│   └── ReservationsController.cs
├── Models/
│   ├── MeetingRoom.cs
│   └── Reservation.cs
├── Services/
│   ├── MeetingRoomService.cs
│   └── ReservationService.cs
├── Settings/
│   └── MongoDBSettings.cs
├── Program.cs
├── appsettings.json
├── appsettings.Development.json
├── launchSettings.json
├── MeetingRoomReservationAPI.csproj
└── MeetingRoomReservationAPI.http
```

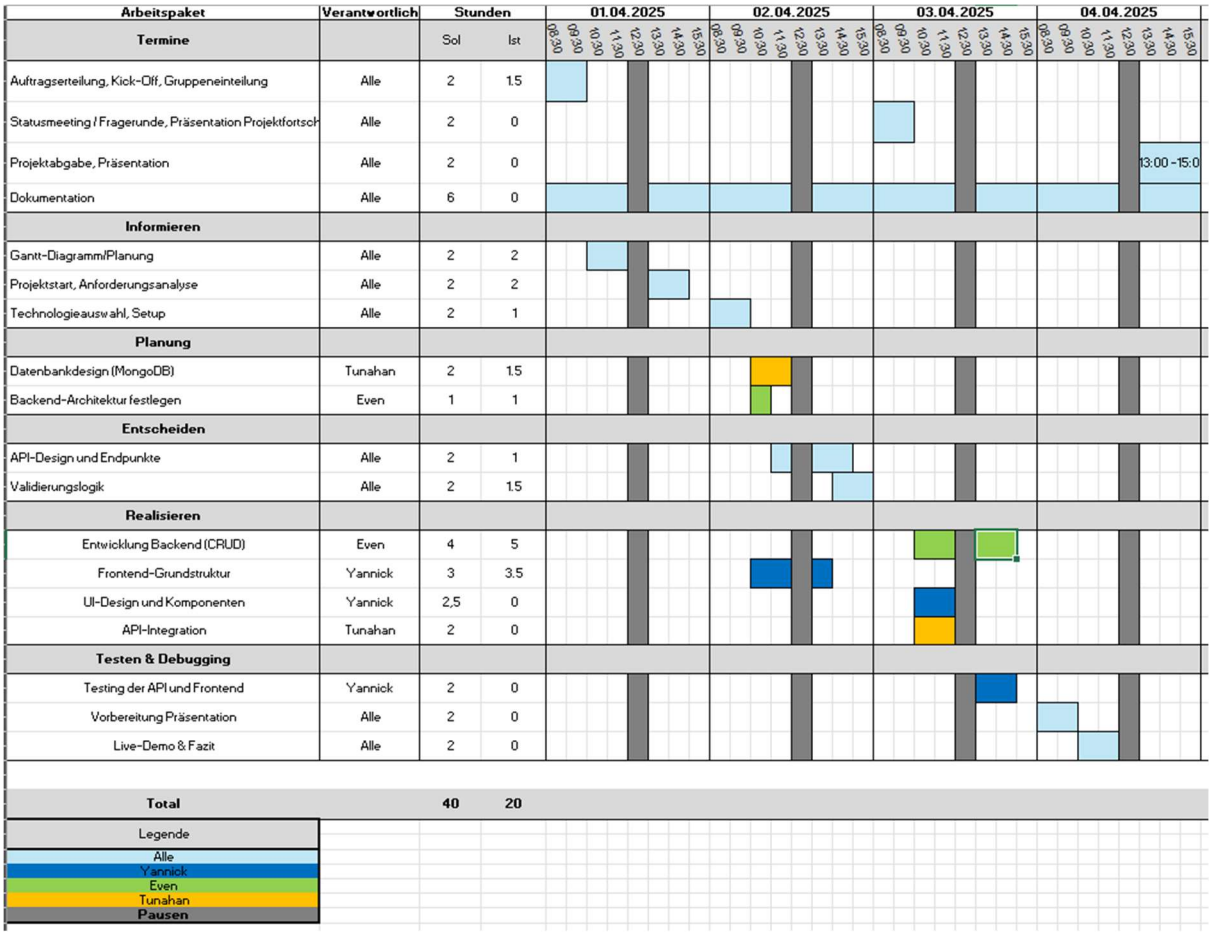
Tests (xUnit)

```
MeetingRoomReservationTests/
├── ReservationServiceTest.cs
└── MeetingRoomReservationTests.csproj
```

5. Projektplanung und Zeitmanagement

- Nutzung von Trello für die Zusammenarbeit
- Definition von Meilensteinen (Backend, Datenbank, Frontend, Testing)
- Erstellung eines Gantt-Diagramms zur Visualisierung der Aufgaben

5.1 GANTT digramm



Arbeitspaket	Verantwortlich	Stunden	
Termine		Sol	Ist
Auftragserteilung, Kick-Off, Gruppeneinteilung	Alle	2	1.5
Statusmeeting / Fragerunde, Präsentation Projektfortsch	Alle	2	0
Projektabgabe, Präsentation	Alle	2	0
Dokumentation	Alle	6	0
Informieren			
Gantt-Diagramm/Planung	Alle	2	2
Projektstart, Anforderungsanalyse	Alle	2	2
Technologieauswahl, Setup	Alle	2	1
Planung			
Datenbankdesign (MongoDB)	Tunahan	2	1.5
Backend-Architektur festlegen	Even	1	1
Entscheiden			
API-Design und Endpunkte	Alle	2	1
Validierungslogik	Alle	2	1.5
Realisieren			
Entwicklung Backend (CRUD)	Even	4	5
Frontend-Grundstruktur	Yannick	3	3.5
UI-Design und Komponenten	Yannick	2,5	0
API-Integration	Tunahan	2	0
Testen & Debugging			
Testing der API und Frontend	Yannick	2	0
Vorbereitung Präsentation	Alle	2	0
Live-Demo & Fazit	Alle	2	0
Total		40	20

5.2 Frontend-Entwicklung

- Entwicklung von UI-Komponenten
- Implementierung von Formularen für Räume und Reservierungen
- Integration der API-Schnittstelle

5.3 Backend-Entwicklung

- Implementierung von CRUD-Endpunkten
- Validierung der Benutzereingaben (z. B. Verhinderung von Reservierungsüberschneidungen)

- Anbindung an MongoDB

5.4 Datenbankintegration

- Design der MongoDB-Modelle
- Implementierung von CRUD-Operationen
- Sicherstellung der Datenintegrität

5.5 Testing & Qualitätssicherung

- Unit-Tests für Backend-Endpunkte
- Manuelles Testen des Frontends
- Fehlerbehebung und Optimierung

6. Abschluss und Abgabe

Die Anwendung wurde gemäss den definierten Anforderungen entwickelt und erfolgreich getestet. Die finale Abgabe umfasst:

- Den vollständigen Quellcode des Projekts in einem Git-Repository
- Eine ausführliche Dokumentation zur Architektur, den verwendeten Technologien und den wichtigsten Funktionen
- Eine detaillierte Beschreibung der Backend- und Frontend-Funktionalitäten
- Eine Gantt-Diagramm-basierte Projektplanung mit einer Analyse des tatsächlichen Zeitaufwands
- Eine Übersicht über die durchgeführten Tests, gefundene Fehler und deren Behebung