Deskflex

Christoph Nie, Leon Rother,
Moritz Mürle, Jannik Filpe,
Hagen Zagola

Inhalt

- Projektvision & Ziel
- Architekturentscheidungen
- Datenbankdesign
- Tech-Stack & Tools
- Design Patterns
- Qualitätssicherung & Testing
- Metriken zur Codequalität
- CI/CD Setup
- Live Demo
- Projektmanagement & Statistiken
- Highlights & Lessons Learned
- Abschluss & Fragen

Projektvision & Ziel

- Flexible Buchung von Arbeitsplätzen in Unternehmen
- Förderung moderner Arbeitsmodelle (z. B. Hybridarbeit)
- Fokus auf Benutzerfreundlichkeit, Übersichtlichkeit & Skalierbarkeit

Architekturentscheidungen

- Architekturstil: Client-Server
- Klare Trennung von Frontend & Backend
- Modularität, Erweiterbarkeit & Skalierbarkeit
- Backend: ASP.NET Core (C#)
- Frontend: Vue.js (JavaScript)
- Styling: TailwindCSS

Datenbankdesign

- Relationale Datenbank: **SQLite**
- Beziehungen: Building → Floor → Desk (über Foreign Keys)
- Tabellen: Buildings, Floors, Desks.
- Optimiert für performante Abfragen nach Standort & Verfügbarkeit

Tech-Stack & Tools

Ziel:
Saubere Trennung von Zuständigkeiten & Wiederverwendbarkeit

Tech-Stack
Frontend: Vue.js, TailwindCSS
Backend: ASP.NET Core (C#)
Datenbank: SQLite
Entwicklungsumgebungen: Visual Studio, VS Code
Versionierung: Git (GitHub)

1. Repository Pattern (für Datenzugriffe im Backend)
2. MVVM (Frontendstruktur mit Vue.js)
3. Dependency Injection (im Backend)

Design Patterns

Qualitätssicherung & Testing

Teststrategie:

- Fokus auf Unit Tests im Backend
- Tests validieren Datenzugriffe und Logik

Vorteile:

- Frühzeitige Fehlererkennung
- Stabilität durch regelmäßige Pipeline-Ausführung

Metriken zur Codequalität

Metrik	Beschreibung	Maßnahmen	Ergebnis
Code Smells	Verdächtige Codestellen	Identifikation & Refactoring durch Reviews und Analyse-Tools	Reduzierung unnötiger Komplexität
Cyclomatic Complexity	Anzahl möglicher Pfade durch Methoden/Logik	Methoden vereinfacht, Kontrollstrukturen überarbeitet	Einfacherer, testbarer Code
Code Duplication	Wiederverwendung von identischem oder ähnlichem Code	Refactoring & Auslagerung in Hilfsmethoden	Weniger Duplikate, verbesserte Wartbarkeit
Test Coverage	Anteil des Codes, der durch Unit Tests abgedeckt wird	Fokus auf kritische Backend-Komponenten	Grundabsicherung wichtiger Logikpfade
Pipeline-Ergebnisse	Automatische Prüfungen bei jedem Commit/Push	CI/CD Setup mit GitHub Actions	Sofortige Rückmeldung bei Build- oder Testfehlern

CI/CD Setup

Pipeline über GitHub Actions:
Automatischer Build bei jedem Push
Automatischer Test bei jedem Push
Artefakterstellung für Deployment

Nutzen:
Zuverlässige Integration & kontinuierliche Qualitätssicherung

Live Demo

Projektmanagement & Statistiken

Tracking & Planung:

- GitHub Issues & Milestones
- Diskussionen im Team via GitHub Discussions

Kennzahlen:

- Anzahl Commits, Issues & Pull Requests
- Übersichtlicher Fortschritt durch Boards

Highlights & Lessons Learned

Stärken:

- Funktionale Web-App mit klarer UX
- Stabiler Tech-Stack & CI/CD

Learnings:

- Hoher Kommunikationsbedarf bei Architektur
- Frühzeitiges Testing zahlt sich aus

Vielen Dank

Christoph Niederer, Leon Rother,

Moritz Mürle, Jannik Filpe,

Hagen Zagola