

W3WI_AM302 - Fortgeschrittene Systementwicklung



Dozent: Prof. Dr. Michael Eichberg
Kontakt: michael.eichberg@dhbw.de, Raum 149B
Version: 24AMA - Feb. 2026

Inhalte 1. Semester

- Entwicklung von verteilten Anwendungen - Grundlagen
 - Linux/Server Handhabung
 - Buildprozesse
 - Versionsverwaltung
 - Container
-

Bereits bekannt sein sollte:

- Programmiersprache Go aus dem ersten Semester
- Programmiersprache Java aus dem zweiten Semester
- Grundkenntnisse in HTML und CSS sollten parallel in der Web-Programmierung gelehrt werden.
Was wird noch gelehrt?
- Datenbanken findet auch parallel statt.
- GitHub ist aus dem ersten Semester bekannt.
Ist GIT als Versionskontrollsystem erklärt worden?

Inhalte 2. Semester

Durchführung eines Entwicklungsprojekts in kleine(re)n Gruppen.

Prüfungsleistung

Prüfungsleistung: Portfolio (insgesamt 120 Punkte)

1. Semester: Kurztest mit 20 Minuten **mit 20 Punkten**

2. Semester: Vorträge, Code Reviews, Ausarbeitungen, Projekt **mit 100 Punkten**

Ablauf

Hintergrund

■ Modul: 55 VL

■ Modul 5 ECTS

1. Semester 16 VL:

- 1. Sep 2025
- 3. Sep 2025
- 15. Sep 2025
- 6. Okt 2025 (20 Minuten Kurztest)

2. Semester: 39 VL

- 18. Feb 2026 von 09:30 bis 12:45 (4VL)
- 25. Feb 2026 von 09:30 bis 12:45 (4VL)
- 4. März 2026 von 09:30 bis 12:45 (4VL)
- 11. März 2026 von 09:30 bis 12:45 (4VL)
- 18. März 2026 von 09:30 bis 12:45 (4VL)
- 25. März 2026 von 09:30 bis 12:45 (4VL)
- 1. April 2026 von 09:30 bis 12:45 (4VL)
- 8. April 2026 von 09:30 bis 12:45 (4VL)
- 15. April 2026 von 09:30 bis 12:45 (4VL)
- 22. April 2026 von 09:30 bis 12:00 (3VL)

Aufgabenstellung 2. Semester

Projektübersicht

Entwickeln Sie ein webbasiertes, responsives Familien-Dashboard, das als zentrale Informationsplattform für Familienmitglieder dient. Die Anwendung soll verschiedene konfigurierbare Widgets bereitstellen und durch ein Rollenkonzept unterschiedliche Zugriffsrechte ermöglichen - ggf. auf Widget-Level.

Anwendungsbeispiele

- Sie möchten - z. B. in der Küche - ein Dashboard auf einem Tablet anzeigen lassen, das den Familienkalender, die Schulpläne der Kinder, den Wochenplan für das Au Pair, das Wetter und eine To-Do-Liste anzeigt.
- Sie möchten, dass alle Familienmitglieder gegenseitig lesenden Zugriff auf den/die Kalender der Familienmitglieder haben; die Kalender werden ggf. in externen Diensten (Google Calendar, Apple Calendar etc.) geführt
- Sie möchten den Schulplan der Kinder als Widget auf dem Dashboard anzeigen lassen
- Ein Au-Pair soll lesenden Zugriff auf ihren/seinen Wochenplan haben, aber keine administrativen Rechte besitzen
- Die To-Do-Liste kann von allen Familienmitgliedern bearbeitet werden
- ...

Anforderungen

Funktionale Anforderungen

Widget-System

- Implementierung eines modularen Widget-Systems mit mindestens 3 der folgenden Widgets:
 - Stundenplan (für Kinder und Au-Pairs) (MUSS)
 - Gemeinsamer Terminkalender (ggf. als Integration externer Kalenderdienste)
 - Wetteranzeige (mit Standortauswahl)
 - To-Do-Liste
 - Notizen/Pinnwand
- Widgets sollen hinzugefügt, entfernt und auf dem Dashboard positioniert werden können
- Jedes Widget muss individuell konfigurierbar sein (z.B. Datenquelle, Darstellungsoptionen)

Rollenkonzept

Familien-Administrator-Rolle:

- Volle Konfigurations- und Verwaltungsrechte
 - Widgets hinzufügen/entfernen/konfigurieren
 - Benutzer verwalten und Rollen zuweisen
(D. h. Registrierung und Authentifizierung von Nutzern durchführen und Nutzer zu Familiengruppen zuweisen.)
 - Widget-Berechtigungen festlegen

Nutzer-Rolle:

- Eingeschränkte Rechte
 - Dashboard ansehen und sein persönliches Layout anpassen
 - Zugriff nur auf freigegebene Widgets
 - Interaktion mit Widget-Inhalten (z.B. Termine einsehen, nicht aber Dashboard umgestalten)

System-Administrator-Rolle:

- Verwaltung der Anwendung als solches
 - Benutzer- und Familiengruppenverwaltung
 - Systemweite Einstellungen und Wartung

Nicht-funktionale Anforderungen

- Responsive Design
 - Optimierte Darstellung für Desktop, Tablet und Smartphone

- Touch-optimierte Bedienung für mobile Endgeräte
- Verteilte Architektur
 - Klare Trennung von Frontend und Backend
 - RESTful API für die Kommunikation
 - Zustandsverwaltung (State Management) im Frontend
 - Datenpersistenz im Backend
- Erweiterbarkeit
 - Modulare Architektur ermöglicht einfaches Hinzufügen neuer Widgets
 - Klar definierte Schnittstellen zwischen Komponenten
 - Plugin-Architektur für Widget-Entwicklung wäre wünschenswert
- Technische Anforderungen

Entwickeln Sie eine verteilte Anwendung mit folgender Architektur:

 - Frontend als Single Page Application (SPA) ggf. unter Einsatz von responsive UI-Frameworks (z.B. Bootstrap, Tailwind CSS)
 - Backend mit RESTful API; der Technologiestack ist frei wählbar (z.B. Node.js/Express, Python/Flask, Java/Spring Boot)
 - Datenbank zur Persistierung (z.B. PostgreSQL, MongoDB, MySQL)
- Architektur-Dokumentation
 - Erstellen Sie Architekturdiagramme (z.B. mit C4-Modell)
 - Dokumentieren Sie Design-Entscheidungen
 - Begründen Sie die Wahl von Technologien und Architekturmustern
- Datenintegration
 - Integration mindestens einer externen API (z.B. Wetter-API)
 - Synchronisation von Kalenderdaten
 - Optional: Import/Export von Daten

◆ Bemerkung

Einsatz von KI-Tools

Nutzung von KI-Assistenten (z.B. GitHub Copilot, ChatGPT, Claude Code oder OpenCode) zur Code-Generierung ist erlaubt.

KI kann zur Unterstützung der folgenden Aufgaben eingesetzt werden:

- Boilerplate-Code generieren
- Code-Optimierung und Refactoring
- Debugging und Fehleranalyse

- Erstellung von Tests

- Dokumentation

Pflichten bei KI-Nutzung

- Jedes Teammitglied muss jeden Teil der Anwendung erklären können
- Die Architektur und Design-Entscheidungen müssen vom Team begründet werden
- Der Technologiestack muss vom Team begründet werden
- Im Projektbericht: Dokumentation, wo und wie, welche KI eingesetzt wurde
- Reflexion über Vor- und Nachteile des KI-Einsatzes im Projekt

Projektergebnisse (abzugeben)

1. Code und Anwendung

- Lauffähige Webanwendung mit allen Kernfunktionen
- README mit Installationsanleitung

2. Architektur- und Entwicklerdokumentation

- System-Architektur mit Diagrammen
- Datenmodell
- REST API-Spezifikation
- Begründung der Technologiewahl

3. Dokument bzgl. KI-Einsatz: Wo wurde welche KI wie eingesetzt. Wie war Ihre Erfahrung.
4. Dokument, das dokumentiert welche Team-Mitglieder welche Teile bearbeitet haben und mit welchem Anteil. **Wenn Sie als Team bewertet werden möchten, dann teilen Sie mir dies bitte mit und vermerken es explizit in dem Dokument.**
5. Die gehaltenen Präsentationen und Code Reviews sind abzugeben.

Zur Verfügung gestellt wird

Zugriff auf einem Server, der aus dem UNI Netz (ggf. mittels VPN) erreichbar ist.

Zugangsdaten stehen im Moodle.

1. Ablauf 2. Semester (Wann passiert was?)

18. Feb 2026 von 09:30 bis 12:45 (4VL)

- Themenvergabe
- Festlegung der Themen, die behandelt werden sollten ((Advanced) JavaScript, RESTful Architectures, verteilte Systeme, Passwortsicherheit, Grundlagen der Kryptographie, Authentifizierung, Frontend-Entwicklung etc.)

25. Feb 2026 von 09:30 bis 12:45 (4VL)

- Präsentation von ausgewählten Themen, die für die Entwicklung der Anwendung relevant sind
- Beantwortung von Fragen

4. März 2026 von 09:30 bis 12:4 (4VL)

- Präsentation von ausgewählten Themen, die für die Entwicklung der Anwendung relevant sind
- Beantwortung von Fragen

11. März 2026 von 09:30 bis 12:45 (4VL)

- jedes Team: Präsentation des Technologiestacks und der grundlegenden Architektur (Level 1 und 2 (ggf. Level 3) des C4-Modells) Dauer: 20 Minuten pro Team

18. März 2026 von 09:30 bis 12:45 (4VL)

- Präsentation von ausgewählten Themen, die für die Entwicklung der Anwendung relevant sind
- Teamindividuelle Beratung

25. März 2026 von 09:30 bis 12:45 (4VL)

- Vorführung des aktuellen Stands der Anwendung durch jedes Team (ca. 10 Minuten pro Team)
- Feedbackrunde und Fragen

1. April 2026 von 09:30 bis 12:45 (4VL)

- Einführung in strukturierte Code Reviews
- Jede Gruppe erhält eine Einführung in den Code zweier anderer Teams (je Partnerteam ca. 45 Minuten)
- Jede Gruppe präsentiert informell ihre allerersten Erkenntnisse/Eindrücke aus dem Code Review (ca. 2-3 Minuten pro Review)
(Die Sessions von jeweils 45 Minuten dienen dazu, dass Sie den Code der anderen Teams kennenlernen und sich auf das Code Review vorbereiten können; die eigentlichen Code Reviews finden außerhalb der Vorlesungszeit statt.)
- Ich stehe für allg. und gruppenspezifische Fragen zur Verfügung.

7. April: 7:00 (Ereignis)

- Abgabe der dokumentierten Code-Review Ergebnisse
 - Abgabe der Präsentationen zu den Code Reviews
 - Abgabe über Moodle als 4 PDF-Dateien mit folgenden Namensschema:
 - CodeReviewDokumentation_TeamX_TeamY.pdf (Code Review von Team X für Team Y)
 - CodeReviewPräsentation_TeamX_TeamZ.pdf (Code Review von Team X für Team Z)
- (X,Y und Z sind durch die jeweiligen Teambezeichner zu ersetzen.)

8. April 2026 von 09:30 bis 12:45 (4VL)

■ Präsentation der Ergebnisse des Code Reviews

10 Minuten pro Code Review; d.h. 20 Minuten pro Team.

15. April 2026 von 09:30 bis 12:45 (4VL) **Online**

- Finale Besprechung der Anforderungen an die Abgaben
- Teamindividuelle Beratung

21. April 2026 7:00 (Ereignis)

- Abgabe aller Projektergebnisse über Moodle

22. April 2026 von 09:30 bis 12:00 (3VL)

Jedes Team: Livevorführung der fertigen Anwendung. Dauer ca. 15 Minuten pro Team. Es müssen insbesondere folgende Abläufe gezeigt werden:

- Registrierung und Anmeldung von Nutzern am System
- Anlegen und Konfigurieren von Widgets durch den Familien-Administrator
- Zuweisung von Rollen
- Persönliche Konfiguration des Dashboards durch einen Nutzer

Teams

- die Teamgröße beträgt 4 Studierende
- die Aufteilung der Aufgaben im Team erfolgt selbstorganisiert, muss aber dokumentiert werden
- Eine grobe Aufteilung entlang der Dimensionen: Frontend, Backend und Widget-Entwicklung ist naheliegend, aber nicht zwingend erforderlich; auf jeden Fall muss *jedes* Teammitglied in der Lage sein die dynamische und statische Architektur zu erklären.

