W3WI_110 - Entwicklung verteilter Systeme

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

Ausgewählte Inhalte gem. MHB - Web-Programmierung

Kerninhalte

- HTML, CSS, JavaScript als clientseitige Web-Technologien und aktuelle APIs (z.B. HTML5 und verwandte Technologien)
- Übertragungsprotokolle und APIs zwischen Client und Server (z. B. HTTP, HTTPS, WebSockets, XMLHttpRequest, Fetch API)
- Kommunikation zwischen einzelnen Komponenten Web-basierter Anwendungen
- Optimierung von Webseiten für verschiedene Zielsysteme

Zusatzinhalte

- Vertiefung von Frameworks
- Dynamische serverseitige Erzeugung von Webseiten

Ausgewählte Inhalte gem. MHB - Verteilte Systeme

- Terminologie, Konzepte, Architekturen, Anforderungsprofile und Architekturmodelle für verteilte Systeme
- Entwurfs- und Implementierungsansätze
- Vergleich unterschiedlicher Middleware-Konzepte
- Synchrone und asynchrone Kommunikation, entfernter Methodenaufruf
- Asynchrone Kommunikation und Messaging-Systeme
- Sicherheitsaspekte in verteilten Systemen

Prüfungsleistung - Portfolio

- das Modul hat 55 VL (mit einem Workload von 150 Stunden insgesamt)
  - Verteilte Systeme hat 22VL
  - Web-Programmierung hat 33VL
- Mehrere Bestandteile:
  - Projekt (max. 80 Punkte)
  - 2 Kurzvorträge (max. 40 Punkte)

1. Vorträge

Vorträge - Themen (Web-Programmierung)

- Die folgenden Vorträge behandeln Themen, die für das Projekt direkt relevant sind.
- Die Vorschläge zu den konkreten Inhalten der Vorträge sind als Anregung zu verstehen.

  Es ist ggf. weder möglich noch sinnvoll alle möglichen Themen abzudecken und kleinere Abweichungen sind ggf. möglich. Es ist Ihre Aufgabe einen in sich geschlossenen Vortrag zu gestalten, der die Grundlagen des Themas vermittelt und die Relevanz für die Projektarbeit klar macht.
- Führen Sie immer in das Thema ein und geben Sie auch praktische Beispiele, damit die Relevanz für die Projektarbeit klar wird. D. h. die Vorträge sollen einen hohen „Hands-on"-Anteil haben.
- Sprechen Sie sich untereinander ab, damit die Vorträge nicht inhaltlich überlappen.

CSS Grid Layout

Personen: 1 Person

Dauer: 10 Minuten

Inhalte:

- CSS Grid Grundlagen und Terminologie
- Responsive Layouts mit Grid
- Grid vs. Flexbox: Wann was verwenden?

CSS Custom Properties und Design Systems

Personen: 1 Person

Dauer: 10 Minuten

Inhalte:

- CSS Variablen und CSS Properties (`@property`) Grundlagen
- Scope und Vererbung
- Performance-Überlegungen
- Design Tokens und konsistente Gestaltung

CSS Animationen und Transitionen

Personen: 1 Person

Dauer: 10 Minuten

Inhalte:

- Grundlagen von Transitions
- Animierte Elemente
- Best Practices für flüssige Animationen
- @keyframes Syntax
- Animation Timing Functions
- Komplexe Animationsabläufe

Responsive Designe

Personen: 1 Person
Dauer: 10 Minuten

Inhalte:

- Mobile-First vs. Desktop-First
- Media Queries für verschiedene Geräte
- Viewport-Units und Container Queries
- Touch vs. Maus-Interaktion

CSS Theming und Farbsysteme

Personen: 2 Personen
Dauer: 20 Minuten

Moderne Color Spaces:

- sRGB, Display P3, Lab, LCH
- color-mix() Funktion
- color-contrast() / contrast-color()
- Barrierefreiheit bei Farbwahl

Light/Dark Themes:

- light-dark() Funktion
- color-scheme Property
- Theme-Switcher implementieren
- CSS Custom Properties für Themes

Moderne UI-Komponenten mit HTML/CSS

Personen: 1 Person
Dauer: 10 Minuten

Inhalte:

- Dialog/Modal für Einstellungen
- *Custom Buttons* und *Controls* (`<button>`, etc.)
- Tooltips und *Notifications* (`popover`, etc.)
- *Loading States* und *Progress Bars*

Grundlagen von modernem JavaScript (ES6+)

Personen: 2 Personen
Dauer: 20 Minuten

ES6+ Grundlegende Features:

- Arrow Functions
- Destructuring
- Spread/Rest Operator
- Default Parameters

ES6+Erweiterte Features:

- Template Literals
- Promises und async/await
- Modules (import/export)
- Array-Methoden (map, filter, reduce)

JavaScript Event Handling

Personen: 1 Person
Dauer: 10 Minuten

Inhalte:

- Event Delegation
- Keyboard Events
- Touch Events für Mobile
- **Event-Loop und Performance**

DOM-Manipulation und Virtual DOM Konzepte

Personen: 1 Person
Dauer: 10 Minuten

Inhalte:

- Effiziente DOM-Updates
- DocumentFragment
- Template-Element
- Shadow DOM Grundlagen
- Performance-Best-Practices

JavaScript State Management

Personen: 1 Person
Dauer: 10 Minuten

Inhalte:

- Lokales State Management
- Immutability
- State Synchronisation Client/Server
- Observer Pattern

JavaScript Error Handling und Debugging

Personen: 1 Person
Dauer: 10 Minuten

Inhalte:

- try/catch Best Practices
- Browser DevTools für Debugging
- Error Boundaries
- Logging-Strategien

Browser Storage APIs

Personen: 1 Person
Dauer: 10 Minuten

Inhalte:

- LocalStorage vs. SessionStorage

- [IndexedDB Grundlagen]
- Privacy-Aspekte

API Design

Personen: 1 Person
Dauer: 10 Minuten

Inhalte:

- HTTP Methoden (GET, POST, PUT, DELETE)
- Status Codes
- Request/Response Format (JSON)
- API-Dokumentation

Fetch API

Personen: 1 Person
Dauer: 10 Minuten

Inhalte:

- Fetch API Grundlagen
- Promises und async/await mit Fetch
- Request/Response Handling
- Error Handling
- CORS-Problematik

Web Security Grundlagen

Personen: 1 Person
Dauer: 10 Minuten

Inhalte:

- XSS (Cross-Site Scripting) Prevention
- CSRF Protection
- Input Validation und Sanitization
- Content Security Policy (CSP)
- Sichere Authentication (JWT, Sessions)
- HTTPS und sichere Verbindungen

WebSockets Grundlagen

Personen: 1 Person
Dauer: 10 Minuten

Inhalte:

- WebSocket Protokoll
- Native WebSocket API

- Connection Lifecycle
- Heartbeats und Reconnection
- Wann WebSockets vs. HTTP

Socket.IO

Personen: 1 Person
Dauer: 10 Minuten

Socket.IO Basics

- Installation und Setup
- Events und Rooms
- Namespaces
- Broadcasting
- Verbindungsmanagement

Express.js Grundlagen

Personen: 1 Person
Dauer: 10 Minuten

Express.js Basics

- Routing
- Middleware-Konzept
- Request/Response Handling
- Static File Serving
- Errorhandling

API-Testing und Dokumentation

Personen: 1 Person
Dauer: 10 Minuten

Inhalte:

- Postman/Thunder Client
- Unit Tests für APIs
- Automatisierte Tests
- Mock-Server/-Daten

Software-Architektur für verteilte Anwendungen

Personen: 1 Person
Dauer: 10 Minuten

Inhalte:

- Client-Server Architektur
- MVC/MVVM Patterns

- Separation of Concerns
- Modulare Struktur
- Code-Organisation

Testing für Web-Anwendungen

Personen: 1 Person
Dauer: 10 Minuten

Inhalte:

- Unit Testing (Jest, Vitest)
- Integration Testing
- E2E Testing Grundlagen
- Mocking und Stubs

Build-Tools und Bundling

Personen: 1 Person
Dauer: 10 Minuten

Inhalte:

- Vite/Webpack Grundlagen
- Module Bundling
- Code Splitting
- Minification und Optimization
- Development vs. Production Builds

Accessibility (a11y) für Spiele

Personen: 1 Person
Dauer: 10 Minuten

Inhalte:

- WCAG Grundlagen
- ARIA Roles und Attributes
- Keyboard Navigation
- Screen Reader Support
- Color Contrast und Lesbarkeit

Aufteilung der Vortragsthemen

ID	Titel
1	CSS Grid Layout
2	CSS Custom Properties und Design Systems
3	CSS Animationen und Transitionen
4	Responsive Design
5	Moderne Color Spaces
6	Light/Dark Themes
7	Moderne UI-Komponenten mit HTML/CSS
8	Accessibility (a11y)
9	ES6+ Grundlegende Features
10	ES6+ Erweiterte Features
11	JavaScript Event Handling
12	DOM-Manipulation und Virtual DOM Konzepte
13	JavaScript State Management
14	Browser Storage APIs
15	API Design
16	Fetch API
17	JavaScript Error Handling und Debugging
18	Web Security Grundlagen
19	WebSockets Grundlagen
20	Socket.IO
21	Express.js Grundlagen
22	Build-Tools und Bundling
23	API-Testing und Dokumentation
24	Software-Architektur für verteilte Anwendungen
25	Testing für Web-Anwendungen

Vorträge - Themen (Verteilte Systeme)

- Die folgenden Vorträge sollen alle konzeptioneller Natur sein und das grundlegende Verständnis für die jeweiligen Themen vermitteln.
- Es wird nicht erwartet, dass Sie tief in die technischen Details gehen oder komplexe Implementierungen präsentieren.

!! Wichtig

Studierende, die Präsentationen innerhalb desselben Blocks halten, müssen sich untereinander abstimmen, um Überschneidungen zu vermeiden und auch sicherstellen, dass der/die vorherige Person die Themen, die für die Präsentation vorausgesetzt werden, auch ausreichend behandelt werden.

!! Wichtig

Wenn Sie bestimmtes Wissen voraussetzen, das von einer anderen Studentin oder einem anderen Studenten behandelt werden soll(te), Sie sich aber nicht sicher sind, ob es ausreichend präsentiert wurde, erstellen Sie ggf. eine Backup-Folie, die das Thema abdeckt. Kennzeichnen Sie die Folie explizit als Backup-Folie mit einem Hinweis in welchem inhaltlichen Block das Thema hätte behandelt werden sollen. Diese Backup-Folie wird dann nicht auf das Zeitlimit angerechnet.

Virtualisierung und Virtualisierungsplatformen

01 Einführung in Virtualisierung & Use Cases

- Was ist Virtualisierung? Historischer Kontext und Motivation
- Unterschiedliche Typen/Ebenen der Virtualisierung
- Zentrale Use Cases: Server Consolidation, Cloud Computing, Development/Testing, Isolation
- Vorteile und Trade-offs

02 Hypervisors – Architecture & Types

- Was ist ein Hypervisor?
- Type 1 (bare-metal) vs. Type 2 (hosted) – architektonische Unterschiede
- Full Virtualization vs. Paravirtualization
- Beispiele und wann welcher Typ eingesetzt wird

03 Virtual Machines – Implementation & Management

- VM-Struktur und Komponenten (virtuelle Hardware, Guest OS)
- VM-Lifecycle: Erstellung, Running, Pause/Resume, Snapshots
- VM-Migration (Konzepte der Live Migration)
- Resource Allocation und Isolation

04 Containers & OS-level Virtualization

- Container-Konzept und Unterschiede zu VMs
- Namespaces und cgroups (konzeptionell)
- Container Images und Layering
- Use Cases und Vergleich mit VMs

05 Memory Virtualization

- Das Address-Translation-Problem (guest virtual → guest physical → host physical)
- Shadow Page Tables
- Hardware-assisted Virtualization (EPT/NPT)
- Memory-Management-Techniken (Overcommitment, Ballooning)

06 Network & I/O Virtualization
- Herausforderungen bei der Virtualisierung von Network- und I/O-Geräten
- Emulated vs. Paravirtualized Devices
- SR-IOV (Single Root I/O Virtualization) und Device Passthrough
- Virtuelle NICs und Network Bridges
- Virtuelle Switches und Network Isolation

Network Protocols

07 QUIC (nur verfügbar, wenn wir ≥ 21 Studierende haben)

08 HTTP/3

09 BitTorrent Protocol (nur verfügbar, wenn wir ≥ 25 Studierende haben)

Modern RPC

10 Protobuf

11 Google RPC

Web-App Security

12 SOP (Same-Origin Policy), CORS (Cross-Origin Resource Sharing) (Grundlagen)

13 CORP / COOP / COEP (Cross-Origin Resource/Opener/Embedder Policies) (nur verfügbar, wenn wir ≥ 23 Studierende haben)

14 CSP (Content Security Policy) und SRI (Subresource Integrity)

Einführung und konkrete Beispiele, wie diese Mechanismen eingesetzt/spezifiziert werden und dabei helfen, Angriffe zu verhindern.

Monitoring & Debugging Distributed Systems

15 Log Aggregation mit besonderem Fokus auf der Korrelation von Log-Einträgen

Leader Election

16 Bully Algorithm und/oder Ring Algorithm

Quorum Systems

17 Majority Voting (d. h. quorum-basiertes verteiltes Rechnen)

Consensus Algorithms and Fault Tolerance

18 Consensus Fundamentals & Problem Definition
- Was ist Consensus und warum ist er in verteilten Systemen schwierig?
- Das FLP-Impossibility-Result (konzeptionelles Verständnis)
- Fehlermodelle: Crash Faults vs. Byzantine Faults
- Safety- vs. Liveness-Eigenschaften
- Motivation aus der Praxis: Replicated State Machines, Distributed Databases

19 (Practical) Byzantine Fault Tolerance
- Wann benötigen wir BFT?
- Moderne Entwicklungen
- Einsatz in realen Systemen

20 Paxos Family

- Grundlegender Paxos-Algorithmus (konzeptioneller Überblick, Rollen: Proposers, Acceptors, Learners)
- Warum Paxos korrekt, aber komplex ist
- Multi-Paxos für praktische Systeme
- Einsatz in realen Systemen

21 Raft – Understandable Consensus

- Motivation
- Leader Election, Log Replication, Safety
- Unterschiede zwischen Raft und Paxos (Design-Philosophie)
- Einsatz in realen Systemen

Eventual Consistency

22 Eventual Consistency und Gossip Protocol

23 CRDTs (Conflict-free Replicated Data Types) (nur verfügbar, wenn wir ≥ 22 Studierende haben)

Distributed File Systems

24 Ceph

25 HDFS (nur verfügbar, wenn wir ≥ 24 Studierende haben)

Themenvergabe

Die Präsentationen müssen in der unten aufgeführten Reihenfolge gehalten werden.

Thema	Nebenbedingung
01. Einführung in Virtualisierung & Use Cases	
02. Hypervisors - Architecture & Types	
03. Virtual Machines	
04. Containers & OS-level Virtualization	
05. Memory Virtualization	
06. Network & I/O Virtualization	
07. QUIC	≥ 21 students
08. HTTP/3	
09. BitTorrent Protocol	≥ 25 students
10. Protobuf	
11. Google RPC	
12. SOP, CORS (Foundations)	
13. CORP / COOP / COEP	≥ 23 students
14. CSP and SRI	
15. Log Aggregation	
16. Leader Election (Bully/Ring Algorithm)	
17. Quorum Systems (Majority Voting)	
18. Consensus Fundamentals & Problem Definition	
19. Byzantine Fault Tolerance (PBFT)	
20. Paxos Family	
21. Raft - Understandable Consensus	
22. Eventual Consistency and Gossip Protocol	
23. CRDTs	≥ 22 students
24. Ceph	
25. HDFS	≥ 24 students

Vorträge - Bewertung (jeweils pro Vortrag)

- 19 Punkte für die eigentliche Präsentation
- 1 Punkte für die Abgabe der unterschriebenen und abgearbeiteten Checkliste
- **Sprechen Sie sich ggf. untereinander ab, damit sich die Vorträge nicht inhaltlich überlappen.**
- Hinweise zur Vortragsgestaltung und Checkliste.

!! Wichtig

- Die Präsentation ist an dem Tag, an dem sie gehalten werden soll bis spätesten 7:00 Uhr in Moodle hochzuladen.
- Die Präsentation muss als PDF hochgeladen werden.
- Die Checkliste muss als PDF hochgeladen werden.

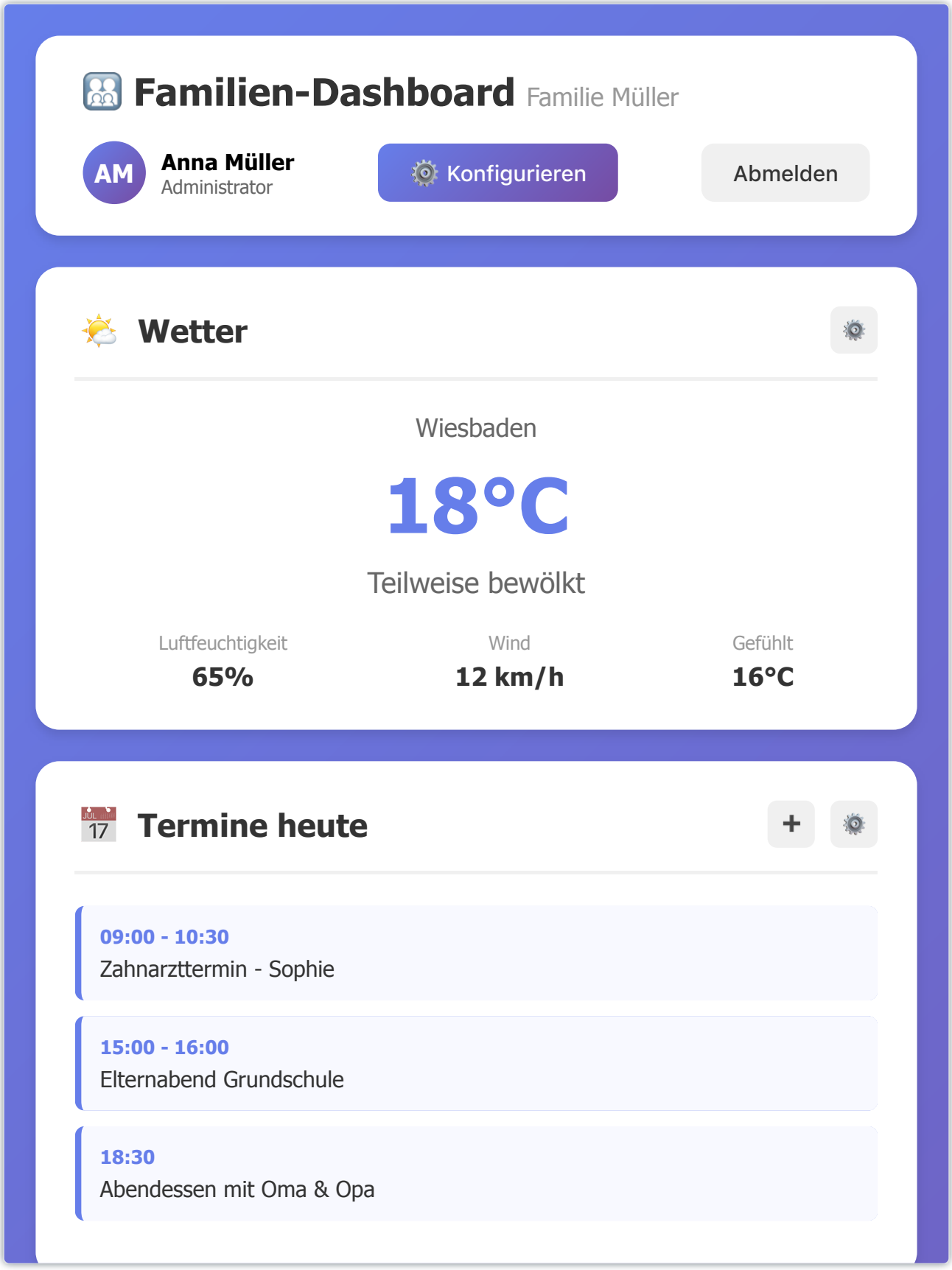
2. Portfolioaufgabe

Projekt/Programmieraufgabe

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.
- ...



Beispiel eines Familien-Dashboards erzeugt mit Claude Code (Sonnet 4.5).

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
  - Widgets sollten Web-Komponenten oder Micro-Frontends (ggf. basierend auf Web-Komponenten) sein, um Wiederverwendbarkeit zu fördern
- 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

Bewertungskriterien für das Projekt

Kategorie	max. 80 Punkte
Architekturpräsentation	max. 05
Code Review	max. 10
Abschlusspräsentation	max. 05
Vorführung	max. 05
Funktionsumfang	max. 15
Dokumentation (Entwickler und Benutzer)	max. 09
Technische Qualität des Videos	max. 01
Qualität des Codes und der Tests (HTML, CSS und JavaScript)[1]	max. 25
Qualität des Buildprozesses[2]	max. 05

!! Wichtig

Es ist ein Dokument einzureichen aus dem hervorgeht:

1. welche KI Tools wofür eingesetzt wurden. (*Fehlanzeige erforderlich!*)
2. wer an welchem Teil mitgewirkt hat. (*Ohne dieses Dokument erfolgt keine Bewertung.*)

---

[1] Es ist neben dem Code auch ein kurzes Video 10 bis max. 15 Minuten einzureichen, dass in die Struktur und die Codebasis einführt. Dieses Video geht in die Benotung ein! Bitte nur im Notfall über Moodle bereitstellen.

[2] Werden Tests ausgeführt und wird am Ende ein Container gebaut?

Projekt - Hinweise

- Ich werde Ihnen Konten auf einem Server bereitstellen, auf dem Sie Ihre Projekte hosten können.
- Das Projekt muss ein Buildscript enthalten, dass es erlaubt das Projekt auf einem standard Ubuntu Server mit einem einzigen Kommando zu bauen.
- Es ist gewünscht, dass Sie Ihr Projekt mit einer expliziten Lizenz veröffentlichen, damit es ggf. auch über die Veranstaltung hinaus von anderen genutzt werden kann.

Gruppennote für Projekt bei Wunsch

Das Projekt ist als Gruppenarbeit ausgelegt und alle Gruppenmitglieder können die gleiche Punktzahl erhalten, wenn dies gewünscht ist.

!! Wichtig

Es ist in jedem Fall zu Protokollieren, wer an welchen Teilen wie mitgewirkt hat. Dieses Protokoll muss am Ende eingereicht werden. Ohne dieses Protokoll erfolgt keine Bewertung!

Das Protokoll wird jedoch nur zur Wertung herangezogen, wenn keine Gruppenbenotung gewünscht ist oder es Unstimmigkeiten gibt.

Gruppenaufteilung

Das Projekt wird von Gruppen mit 4 Studierenden bearbeitet. (Sollte es nicht aufgehen, dann können auch Gruppen mit 3 oder 5 Studierenden gebildet werden.)

Ablauf - W3WI-110 - Entwicklung verteilter Systeme 24SEA

Block 1

- 20. Feb 2026 von 13:00 bis 17:00 - Themenvergabe und Vorlesung
- 27. Feb 2026 von 13:00 bis 17:00 - Vorlesung
- 06. Mar 2026 von 13:00 bis 17:00 - Vorträge (Web) 1. - 12.
- 13. Mar 2026 von 13:00 bis 17:00 - Vorträge (Web) 13. - 25.
- 20. Mar 2026 von 13:00 bis 17:00 - Vorträge (Verteilte) 1. - 12.
- 27. Mar 2026 von 13:00 bis 17:00 - Vorträge (Verteilte) 13. - 25.

Block 2

- 10. Apr 2026 von 13:00 bis 17:00 - Präsentation des aktuellen Projektstands - insbesondere der grundlegenden Architektur (15 Minuten pro Gruppe)
- 14. Apr 2026 von 13:00 bis 17:00 - Code Reviews - allg. Einführung (ca. 45 Minuten); Zuteilung der Reviews; gegenseitige Einführung in den Code (2 * 45 Minutenb)
- 17. Apr 2026 von 13:00 bis 17:00 - Code Reviews - Präsentation der Ergebnisse
- 20. Apr 2026 von 13:30 bis 17:30 **ONLINE** - Besprechung der finalen Abgaben; gruppenindividuelle Betreuung
- 24. Apr 2026 von 13:00 bis 17:00 - Präsentation und Vorführung der Projekte

Code Reviews

14. Apr 2026:

- jede Gruppe führt ein Code Review eines anderen Projektes durch

  (D. h. jede Gruppe führt eine andere Gruppe in ihr Projekt ein, damit die Reviewer die Codebasis verstehen und dann einen Bericht mit konstruktiven Vorschlägen erstellen können - dies geschieht im Nachgang. Die Zuteilung wird am 14. Apr 2026 bekannt gegeben.)
- alle Teile sollen einem Review unterzogen werden: Frontend, Backend, Buildscripte, Projektstruktur, Dokumentation, ...

16. Apr 2026 - 17:00:

Abgabe der Code Review Berichte und Präsentation in Moodle; Übermittlung der Berichte an die jeweiligen Gruppen.

17. Apr 2026:

Die *wichtigsten Ergebnisse werden präsentiert*; es können ggf. auch Erkenntnisse für das eigene Projekts dargestellt werden.