Weiterbildung gibb

HF Informatik & Telekommunikation
Höhere Fachschule

MODUL WEB2
WEB-ENGINEERING 2

Nicolas Dumermuth

Inhaltsverzeichnis

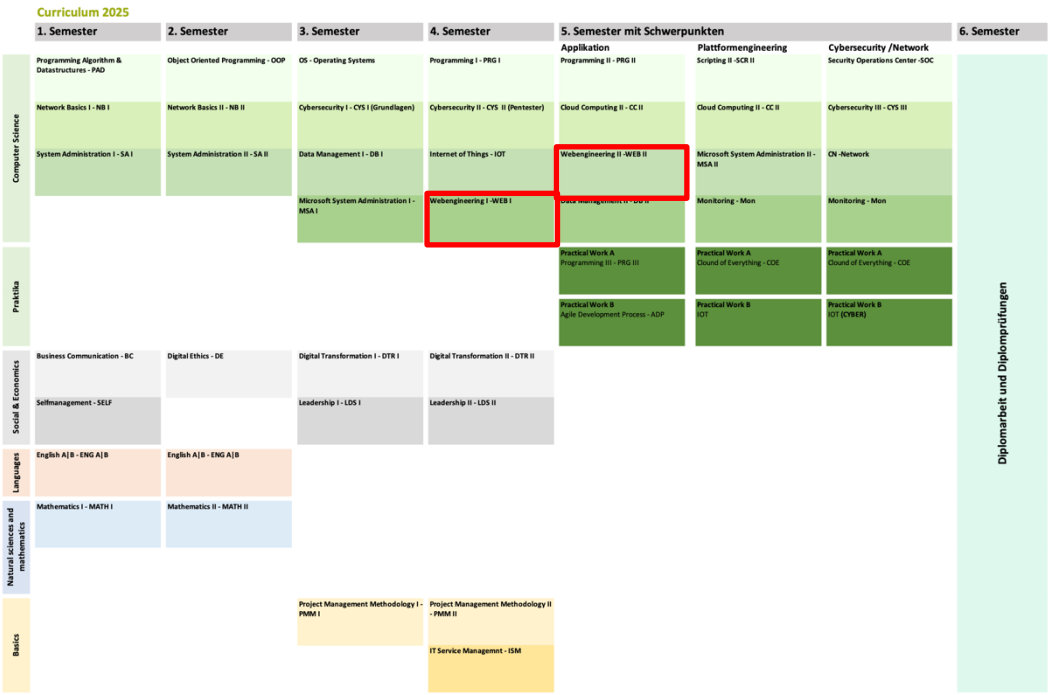
Modulleitfaden

1. Verortung im Curriculum

Das Modul WEB2 baut direkt auf WEB1 auf und vertieft die Entwicklung moderner Single-Page-Applications (SPA) mit Vue.js. Es richtet sich primär an die Fachrichtung Applikationsentwicklung.

WEB1 vermittelt die Grundlagen: HTML5, CSS3, JavaScript (ES6+), DOM-Manipulation, asynchrone Programmierung und API-Integration.

WEB2 überträgt diese Kenntnisse auf Vue.js: Komponentenbasierte SPAs mit Composition API, Vue Router, Pinia für State-Management und RESTful API-Integration.



2. Umfang und Lernstunden

Lernform	Anzahl Lernstunden	Bemerkungen
Kontaktstudium	48	Seminare, Workshops
Angeleitetes Selbststudium	30	Projektarbeit «SPA-Entwicklung»
Individuelles Selbststudium	8	Individuelle Vertiefung, Lernvideos
Total Lernstunden	**86**	

3. Kompetenzen

3.1 Fachliche Kompetenzen

- **Vue.js Grundlagen**
  - Entwicklung von Single-File Components (SFCs) mit Template, Script (Composition API) und Styles
  - Professioneller Einsatz der Composition API: ref, reactive, computed, watch, Lifecycle Hooks
  - Verständnis des Reaktivitätssystems und der Template-Syntax mit Direktiven (v-if, v-for, v-model)
- **Komponentenarchitektur**
  - Planung modularer, wartbarer und skalierbarer Frontend-Architekturen
  - Kommunikation zwischen Komponenten mit Props und Custom Events (Props Down, Events Up)
  - Strukturierung von Container- und Präsentationskomponenten nach Best Practices
- **Routing und Navigation**
  - Implementierung von Client-Side Routing mit Vue Router
  - Konfiguration dynamischer Routen, Named Routes und Navigation Guards
  - Entwicklung mehrschichtiger Navigationsstrukturen für komplexe SPAs
- **State-Management**
  - Zentrales State-Management mit Pinia für komplexe Anwendungszustände
  - Unterscheidung zwischen lokalem Komponenten-State und globalem Application-State
  - Implementierung von Actions, Getters und reaktivem State
- **API-Integration und Tooling**
  - Asynchrone Datenflüsse mit RESTful APIs (Fetch/Axios), Promises und async/await
  - Professioneller Einsatz moderner Build-Tools (Vite) und des Vue-Tooling-Ökosystems
  - Anwendung von Vue DevTools, ESLint, Prettier und GitLab für Code-Qualität

3.2 Überfachliche Kompetenzen

- **Methodenkompetenzen**
  - Systematische Planung und Dokumentation von SPA-Projekten
  - Analyse und Behebung komplexer Frontend-Probleme mit Vue DevTools
  - Prozessorientiertes Denken bei der Entwicklung komponentenbasierter Architekturen
  - Effektive Nutzung der Vue-Dokumentation und modernen Entwicklungswerkzeugen
  - Selbstgesteuertes Lernen durch iterative Übungen und Projektarbeit
- **Sozialkompetenzen**
  - Kommunikationsfähigkeit beim Präsentieren von SPA-Projekten im Fachgespräch
  - Teamfähigkeit in Projektarbeiten mit Code-Reviews und gemeinsamem Troubleshooting
  - Konstruktive Feedback-Kultur bei Architektur- und Implementierungsentscheidungen
- **Selbstkompetenzen**
  - Eigenverantwortung beim selbstständigen Durcharbeiten der Vue-Übungen
  - Flexibilität im Umgang mit dem dynamischen Vue.js-Ökosystem
  - Ethisches Handeln mit Bewusstsein für Datenschutz, Accessibility und nachhaltige Webentwicklung

4. Vorkenntnisse

Das Modul baut auf den Grundlagen aus **PRG1** (Programmierung 1) und **WEB1** (Web-Engineering 1) auf. Fundierte Kenntnisse aus WEB1 sind zwingend erforderlich und werden im Modul direkt angewendet.

Konkret erforderlich aus WEB1:

› **HTML5 & CSS3**: Semantisches HTML, CSS3-Layoutmodelle (Flexbox, Grid), Responsive Design mit Mobile-First-Ansatz
› **JavaScript ES6+:** Arrow Functions, Template Strings, Destructuring, Spread/Rest-Operator, Klassen
› **Asynchrone Programmierung**: Promises, async/await, Fetch API für API-Integration
› **DOM-Manipulation**: Event-Handling, dynamisches Rendering von Inhalten
› **Tooling**: Git/GitLab für Versionskontrolle, Branches, Merge-Requests

Empfohlen:

› Objektorientierte Programmierung mit Klassen, Objekten und Vererbung
› Grundverständnis von REST-APIs, HTTP-Requests, JSON und HTTP-Status-Codes
› Erfahrung mit Browser-DevTools für Debugging

Vue.js-spezifische Vorkenntnisse sind nicht erforderlich – das Framework wird im Modul von Grund auf vermittelt. Sollten dennoch Wissenslücken in den WEB1-Vorkenntnissen bestehen, sind diese vor Modulstart eigenständig aufzuarbeiten.

5. Hinweise zum Selbststudium

Das Modul kombiniert strukturierte Präsenzphasen mit intensivem Selbststudium und einem umfangreichen SPA-Projekt. Alle Unterlagen sind so konzipiert, dass sie sowohl im Präsenzunterricht als auch im Selbststudium funktionieren. Bei weiterführenden Links (z.B. Vue-Dokumentation, Vue Mastery) können Sie individuell entscheiden, ob die Inhalte bereits bekannt sind oder noch vertieft werden müssen.

Im Präsenzunterricht demonstriert der Dozierende praktische Implementierungen mit Vue.js, Vite und modernen Entwicklungswerkzeugen. Die Studierenden replizieren diese in ihren eigenen Entwicklungsumgebungen (Windows, Linux oder macOS). Dabei entsteht ein gemeinsamer Lernprozess mit Live-Coding, Troubleshooting mit Vue DevTools und Diskussionen über Best Practices.

Die Übungen (z.B. ToDoApp, Kanban-Board) im GitLab-Repository werden in Eigenverantwortung abgeschlossen. Challenges können einzeln, im Team und/oder mit Hilfe von KI-Tools gelöst werden. Der Output steht im Fokus – wichtig ist aber, dass Sie die Lösungen immer verstehen und nachvollziehen können.

Das angeleitete Selbststudium konzentriert sich auf die Projektarbeit SPA-Entwicklung (30 Lektionen), in der die Studierenden eine vollständige Single-Page-Application entwickeln. Diese Arbeit erfolgt in Gruppen (2-3 Personen) mit klaren Meilensteinen: Projekteingabe, Code-Entwicklung, Dokumentation und Livedemonstration. Die Projektarbeit ermöglicht individuelle Vertiefung in selbst gewählte Schwerpunkte.

6. Bezug zu RLP Handlungskompetenzen

Die Grundlage für die nachfolgenden Inhalte bildet der Rahmenlehrplan HF Informatik. Er legt die zentralen Bildungsziele, Kompetenzen und Schwerpunkte der Ausbildung an höheren Fachschulen fest und dient damit als verbindlicher Bezugspunkt für die Gestaltung des Unterrichts.

https://www.becc.admin.ch/becc/public/bvz/beruf/show/321

Kompetenz (RLP-Ref.)	Beschreibung	Niveau (1–4)	Konkretisierung im Modul
A2.5	Informations- und Kommunikationstechnologien (ICT) professionell einsetzen und etablieren.	3	Professioneller Einsatz moderner Frontend-Technologien: Vue.js als reaktives Framework, Vite als Build-Tool, Vue Router für Navigation und Pinia für State-Management. Kommunikation mit Backends über REST-APIs mit Fetch/Axios, JSON-Formate. GitLab für Versionskontrolle und Vue DevTools zur Fehlersuche und State-Inspektion.
A3.3	Neue Technologien kritisch reflexiv beurteilen, adaptieren und integrieren	3	Kritischer Vergleich von Vue.js mit React und Angular anhand von Lernkurve, Performance und Community-Support. Evaluation von Trade-offs: Options API vs. Composition API, Single-File Components vs. Separation of Concerns, verschiedene State-Management-Lösungen. Studierende begründen ihre Entscheidungen im Anforderungskontext.
A3.4	Die eigenen digitalen Kompetenzen kontinuierlich weiterentwickeln	3	Selbstgesteuertes Lernen durch Vue-Dokumentation, Code-Beispiele im GitLab-Repository und Video-Tutorials (Vue Mastery). Iterative Übungen (ToDoApp, Kanban-Board) und die Projektarbeit fördern kontinuierliche Weiterentwicklung. Livedemonstration erfordert Reflexion von Architekturentscheidungen und tiefes Technologieverständnis.

B4.2	ICT-Problemstellungen unter Berücksichtigung vernetzten Denkens, Entwicklung neuer ICT-Lösungen und Anwendung aktueller Technologien identifizieren, analysieren und lösen	3	Analyse komplexer SPA-Anforderungen wie Kanban-Boards mit Drag-and-Drop oder Blogs mit dynamischen Routen. Überführung in komponentenbasierte Architektur. Lösung typischer SPA-Herausforderungen: Client-Side Routing, asynchrones Datenladen, State-Synchronisation zwischen Komponenten durch Vue-Patterns (Props Down, Events Up, Composables).
B4.6	Applikationen unter Beachtung übergeordneter Konzepte wie der ICT-Strategie, Standards etc. in die Softwarearchitektur integrieren.	3	Professionelles Vue.js-Setup: Vite als Build-Tool mit Hot Module Replacement, Vue DevTools für Komponenten- und State-Inspektion, ESLint und Prettier für Code-Qualität, GitLab für Versionskontrolle. VS Code oder WebStorm mit Volar Extension. Node.js und npm für Paketmanagement.
B5.1	ICT-Projekte oder Vorhaben eigenständig bis zur Ausführungsreife planen und steuern.	3	Projektarbeit erfordert eigenständige Planung: Definition von User Stories, Strukturierung der Komponentenhierarchie, Planung der State-Management-Architektur. Koordination der Aufgabenverteilung in Gruppen mit GitLab-Issues, Tracking des Fortschritts, termingerechte Abgabe mit vollständiger Dokumentation.
B10.1	Die Architektur der Software bestimmen und die Entwicklung unter Berücksichtigung von Betrieb und Wartung planen und dokumentieren	3	Entwurf komponentenbasierter Vue-Architekturen: Trennung zwischen Container- und Präsentationskomponenten, klare Komponentenhierarchie mit definierten Verantwortlichkeiten, durchdachte State-Management-Strategie (lokaler vs. globaler State mit Pinia). Dokumentation über README, Code-Kommentare, Architekturdiagramme im GitLab-Repository.

B10.2	Applikationen unter Beachtung übergeordneter Konzepte wie der ICT-Strategie, Standards etc. in die Softwarearchitektur integrieren	3	Integration von Vue-SPAs in bestehende System-Landschaften: Anbindung von REST-APIs, Implementierung von JWT-Authentifizierung, Einhaltung von Web-Standards (HTML5 Semantic, ARIA, Responsive Design). Projektarbeit erfordert Backend-Integration mit CORS-Konfigurationen, HTTP-Status-Codes und robustem Error-Handling.
B11.1	Vorgaben für die Konzipierung eines Softwaresystems mit einer formalen Methode analysieren.	3	Analyse von Anforderungen durch Wireframes in Figma, User Stories und Komponentendiagramme zur Visualisierung der Hierarchie. Data-Flow-Analysen zur Bestimmung, welche Komponente welchen State hält. Überführung von Anforderungen (Drag-and-Drop, Task-Filter) in konkrete Vue-Komponenten und Pinia Store-Actions.
B11.3	Spezifikation in einer geeigneten Programmiersprache umsetzen	3	Umsetzung von Spezifikationen mit Vue.js Composition API: Template-Direktiven (v-if, v-for, v-model), reaktive Referenzen (ref, reactive, computed, watch), Lifecycle Hooks (onMounted), wiederverwendbare Composables. Am Beispiel ToDoApp: gefilterte Listen, Formular-Handling, API-Calls mit async/await, persistierter State mit localStorage.
B11.5	Mobile und verteilte Applikationen unter Berücksichtigung zeitgemässer Architekturmuster bzw. Referenzarchitekturen implementieren	3	Moderne SPA-Architekturmuster: Responsive Design mit Mobile-First-Ansatz (CSS Grid, Flexbox, Media Queries), Client-Side Routing für nahtlose Navigation, asynchrones Laden von Daten mit Fetch API. Projektarbeit erfordert vollständig responsive SPA, die auf verschiedenen Geräten funktioniert und asynchron mit Backend-Services über REST-APIs kommuniziert.
B11.6	Prinzipien, Methoden und Werkzeuge für die arbeitsteilige Entwicklung und Anwendung von umfangreichen Softwaresystemen zielorientiert bereitstellen und systematisch umsetzen	3	Testbarkeit durch saubere Architektur: Auslagerung von Logik in Composables, die unabhängig von der UI testbar sind. State-Manipulationen als reine Funktionen. Manuelles Testing von User-Flows, Nutzung von Vue DevTools zur Fehlersuche. Vorbereitung auf zukünftiges automatisiertes Testing mit Vitest oder Jest.

7. Semesterplan

KW	Datum	Inhalt/ Thema	Anzahl L.	Lernform	Sozialform	Lernort	Abgaben
KW 6	07.02.2026	**Intro & Setup** Grundlagen, Vite, Single-File Components, Composition API, Einführung Vue.js	6L	K/AS	EA/PA	Remote	
KW 7	14.02.2026	**SPA-Essentials & Komponenten-Design** Komponenten-Design, Props, Events, Computed Properties, Vue-Router, State-Management	6L	K/AS	EA/PA	Präsenz	
KW 8	21.02.2026	**SPA-Vertiefung** Refresher W1/W2, Vorbereitung Prüfung, Reserve	6L	K/AS	EA/PA	Remote	
KW 9	28.02.2026	**Prüfung LB1 (Vue-Basics)** Projekteingabe/Gruppenbildung (LB2), Kickoff-Projekte	6L	K/AS/IS	EA/GA	Präsenz	**LB1: Prüfung, LB2: Projekteingabe**
KW 10	07.03.2026	**Projektarbeit** Zwischengespräche	6L	AS/IS	PA/GA	Präsenz	
KW 11	14.03.2026	**Projektarbeit** Zwischengespräche	6L	AS/IS	PA/GA	Präsenz	
KW 12	21.03.2026	**Abschluss Projektarbeit** Zwischengespräche	6L	AS/IS	PA/GA	Präsenz	**LB2: Code + Kurzdokumentation**
KW 13	28.03.2026	**Abschluss Modul** Livedemonstrationen, Reserve, Abschluss Modul	6L	K/AS/IS	PA/GA	Remote	**LB2: Livedemonstration**

Lernform: K (Kontaktstudium); AS (Angeleitetes Selbststudium); IS (Individuelles Selbststudium)
Sozialform: EA (Einzelarbeit); PA (Partnerarbeit); GA (Gruppenarbeit)

8. Arbeits- und Leistungsnachweise

Bereich	Kriterien	Bewertung/ Gewichtung
LB1: Praktische Prüfung Einzelarbeit	Die Studierenden lösen praktische Aufgaben zu Vue.js-Grundlagen in einer 90-minütigen Prüfung. Geprüft werden Kenntnisse zu Single-File Components, Composition API, Props, Events, Computed Properties, Vue-Router und State-Management.	Note (50%)
LB2: SPA-Projekt Einzel- oder Gruppenarbeit (2 bis 3 Personen)	**Projekteingabe**: Die Studierenden beschreiben ihre SPA-Idee, definieren SMART-Ziele und Features, wählen Technologien und erstellen ein GitLab-Repository. **Kurzdokumentation**: Die Studierenden erstellen eine Dokumentation (4 bis 6 Seiten) mit Anforderungen und Zielerreichung, technischer Umsetzung (Architektur, Komponenten, Routing, State-Management), Visualisierungen und Fazit/Lessons Learned. **Technische Umsetzung**: Die Studierenden entwickeln eine funktionsfähige, responsive SPA mit durchdachter Komponentenarchitektur. Sie implementieren Client-Side Routing, setzen State-Management um und zeigen sauberen Code mit nachvollziehbarer Git-Historie. Bewertet werden Codestruktur, Komponentenarchitektur, Routing, State-Management, Funktionalität, UI/UX Design, Responsive Design und Git-Workflow. **Livedemonstration**: Die Studierenden präsentieren ihre entwickelte SPA in einer Live-Demonstration (15 bis 20 Minuten, Remote via Teams). Sie zeigen die wichtigsten Funktionen, erläutern technische Entscheidungen und beantworten Fachfragen. Bewertet werden Präsentation, Vortragsweise und technisches Verständnis.	Note (50%)

9. Modulspezifische Hinweise

9.1 Entwicklungsumgebung

Das Modul ist plattformunabhängig aufgebaut. Studierende arbeiten auf eigenen Geräten mit Windows, macOS oder Linux. Empfohlene Entwicklungsumgebung:

› **Editor**: VS Code mit Volar Extension (offiziell empfohlen für Vue 3) oder WebStorm
› **Browser**: Chrome oder Firefox mit Vue DevTools (Browser-Extension)
› **Node.js**: Version 22+ für Vite und npm
› **Versionskontrolle**: GitLab unter git.gibb.ch
› **Design/Prototyping**: Figma (kostenloser Account)
› **Code-Qualität**: ESLint und Prettier (automatisch im Vite-Setup enthalten)

9.2 Repository-Struktur

Das persönliche GitLab-Repository folgt der Struktur WEB2_Vorname_Nachname mit Unterordnern:

› ***/unterlagen***: Kursunterlagen, Slides, Cheat Sheets
› ***/uebungen***: ToDoApp, Kanban-Board, weitere Challenges
› ***/projekt***: SPA-Projektarbeit mit README, Mockups, Code

Studierende geben das Repository für den Dozierenden mit Maintainer-Rolle frei, arbeiten mit regelmässigen aussagekräftigen Commits und nutzen Feature-Branches für die Entwicklung (z.B. feature/user-auth, feature/drag-drop).

9.3 Projektthemen

Mögliche SPA-Projekte (40-60 Lektionen pro Gruppe, 2-3 Personen):

› **Quiz-App**: Kategorien, Highscore, API-Integration (Open Trivia DB)
› **Wetter-App**: Standortsuche, Forecast, Diagramme (OpenWeatherMap API)
› **Ernährungstracker**: Kalorienzähler, Rezepte, Visualisierung (Nutritionix API)
› **Umfrage-App**: Erstellung, Abstimmung, Ergebnis-Visualisierung (Diagramme)
› **Sprachlern-App**: Karteikarten, Spaced Repetition, Fortschritt-Tracking
› **Projektmanagement-Tool**: Kanban-Board, Drag-and-Drop, Task-Filter

Eigene Projektideen sind nach Rücksprache mit dem Dozierenden möglich. Wichtig: Ausreichende Komplexität für State-Management, Routing und API-Integration.

9.4 Ressourcen

Die offizielle Dokumentation von Vue.js, Vue Router, Pinia und Vite bildet die zentrale Lerngrundlage. Ergänzend stehen Tutorials auf Vue Mastery und Vue School sowie die Community über das Vue.js Forum und Discord zur Verfügung.