A.3.1. Softwareentwicklung (Letztentwurf)

Die Kernkompetenzen und die transversalen Kompetenzen bilden das zentrale Ziel des Unterrichtsgegenstandes und sollen den für diesen Unterrichtsgegenstand zugedachten Beitrag zur Erreichung der allgemeinen Bildungsziele (Berufsfähigkeit, Studierfähigkeit & lebenslanges Lernen, mündiger Bürger, Lebensgestaltbarkeit) und der speziellen Bildungsziele (Entrepreneurship, Digitale Applikation & KI, Nachhaltigkeit, Future Skills) leisten.

Bereich	**P. Planung und Entwurf**	**I. Problemlösung und Implementierung**	**Q. Qualitätssicherung, Auslieferung und Betrieb**	**E. Entwicklungsprozess und Softwareprojekte**
Kern-kompetenz	Software planen und entwerfen	Software entwickeln	Softwarequalität sicherstellen sowie Software ausliefern und betreiben	Entwicklungsprozess und Softwareprojekte organisieren, unterstützen und umsetzen
Transversale Kompetenz	▪ Analytisches Denken ▪ Kreativität und Innovationskraft ▪ Vernetztes, multiperspektivisches Denken	▪ Selbstständiges Problemlösen ▪ Selbstwirksamkeit, Fokussierung und Zielstrebigkeit ▪ Resilienz (Ausdauer, Stressresistenz, Anpassungsfähigkeit, Flexibilität, Agilität)	▪ Kritisches Denken ▪ Achtsamkeit ▪ Verantwortungsbewusstsein und Zuverlässigkeit	▪ Selbstorganisation und digitales Projektmanagement ▪ Kollaboration, Kooperations- und Teamfähigkeit ▪ Prozessorientierung
I. Jahrgang				
Kompetenz	Schülerinnen und Schüler können P1.1 analytisch und strukturiert denken. (H) P1.2 Anforderungen für Software spezifizieren. (H) P1.3 Problemstellungen analysieren und Lösungsansätze entwickeln. (O/H)	Schülerinnen und Schüler können I1.1 grundlegende Konzepte der strukturierten Programmierung anwenden. (H) I1.2 grundlegende Funktionen von integrierten Entwicklungsumgebungen (IDEs) nutzen. (H) I1.3 Syntaxfehler verstehen und beheben. (H)	Schülerinnen und Schüler können Q1.1 Ausführungsumgebungen für Programme erklären sowie Programme übersetzen und ausführen. (O/H) Q1.2 Codes lesen und verstehen sowie die Ausführung von Programmen analysieren und Fehler beheben. (H) Q1.3 die Erfüllung von Anforderungen prüfen. (H)	Schülerinnen und Schüler können E1.1 Programmiertätigkeit an einem einfachen Softwareentwicklungszyklus ausrichten. (O/H) E1.2 Softwareprojekte mit wirtschaftlichem Anwendungsbezug planen, umsetzen und präsentieren. (H)
Lehrstoff	P1.1 Training des analytischen und strukturierten Denkens P1.2 funktionale und nicht funktionale Anforderungen P1.3 Techniken der Problemanalyse (z. B. Aspekte des Computational Thinkings wie Generalisierung, Abstraktion, Dekomposition, Mustererkennung); Techniken der algorithmischen Problemlösung (z. B. Divide and Conquer, Brute Force); allg. Vorgehensweisen (z. B. ideen- bzw. hypothesengeleitete sowie inkrementelle Vorgehensweise, Orientierung am Problemlösungszyklus); Algorithmisierung mit Grundbausteinen der strukturierten Programmierung; Darstellungsformen für Algorithmen wie z. B. Pseudocode, Struktogramme, Flussdiagramme	I1.1 Variablen, Datentypen, Operatoren, Ausdrücke, Kontrollstrukturen, Modularisierungstechniken (wie Funktionen, Prozeduren, Methoden), Arrays und Listen, Input, Output, Basiskonzepte der Objektorientierung; Formatierung von Code etc. I1.2 (didaktische) Funktionen zur Unterstützung des Lernprozesses, IDE-Projektorganisation, Shortcuts, Auto-Format, Auto-Save, Codenavigation, Themes und Schriften, Suchen und Ersetzen, Hilfen für Code-Lesen / Code-Verstehen / Code schreiben / Code ändern etc. I1.3 Syntax von Computerprogrammen; Interpretation und Beheben von Syntaxfehlern	Q1.1 Development-Kits und Ausführungsumgebungen (z. B. für kompilierte, interpretierte oder JIT-kompilierte Sprachen) Q1.2 Semantik von Code; Code-Tracing-Techniken wie z. B. Print-Statements, Logging, Nutzung von Debuggern (Haltepunkte, Debugger-Fenster, Step-Actions, Variablenwerte); Laufzeit- und Semantikfehler Q1.3 Verifikation von funktionalen und nicht funktionalen Anforderungen (aufbauend auf A 1.2)	E1.1 Codieren und kompilieren – Syntaxfehler beheben – ausführen und testen – Laufzeit- und Semantikfehler beheben E1.2 Anwendung der Kernkompetenzen aus den Kompetenzbereichen des 1. Jahrganges

II. Jahrgang: 3. und 4. Semester				
Kompetenz	**3. Semester**: Schülerinnen und Schüler können P2.1 objektorientierte Analysen durchführen. (O/H)	**3. Semester**: Schülerinnen und Schüler können I2.1 Konzepte der objektorientierten Programmierung anwenden. (H) I2.2 Ausnahmebehandlungen einsetzen. (H) I2.3 Bibliotheken nutzen. (H) I2.4 fortgeschrittene Funktionen von IDEs nutzen. (H)	**3. Semester**: Schülerinnen und Schüler können Q2.1 fortgeschrittene Debugger-Funktionen anwenden. (H) Q2.2 Codedokumentation finden, verstehen und erstellen. (H) Q2.3 Coding-Standards einhalten. (H)	**3. Semester**: Schülerinnen und Schüler können E2.1 grundlegende Funktionen eines Versionskontrollsystems (VCS) nutzen. (H)
	4. Semester: Schülerinnen und Schüler können P2.2 Kundenanforderungen ermitteln und dokumentieren. (H)	**4. Semester**: Schülerinnen und Schüler können I2.5 weitere Konzepte der objektorientierten Programmierung anwenden. (H) I2.6 Algorithmen und Datenstrukturen auswählen und anwenden. (H) I2.7 grafische Benutzeroberflächen implementieren. (H)	**4. Semester**: Schülerinnen und Schüler können Q2.4 die Laufzeitkomplexität von Programmen abschätzen. (H) Q2.5 die Erfüllung von Kundenanforderungen nachweisen und dokumentieren. (H)	**4. Semester**: Schülerinnen und Schüler können E2.2 weitere Funktionen eines VCS nutzen. (H) E2.3 Softwareprojekte mit wirtschaftlichem Anwendungsbezug planen, umsetzen und präsentieren. (H)
Lehrstoff	**3. Semester:** P2.1 Grundlegende Methoden (z. B. Textanalyse, CRC-Karten); Strukturelle Modellierung (z. B. Klassendiagramme, Objektdiagramme); Dynamische Modellierung (z. B. Sequenzdiagramme, Zustandsdiagramme); Dokumentation mit Auszeichnungssprachen (z. B. AsciiDoc, Markdown) und Diagrammen (Diagramm-Werkzeuge, Diagrams as Code)	**3. Semester:** I2.1 Klassen, Objekte, Datenfelder, Konstruktoren, Methoden, Enums; Zugriffsmodifikatoren und Kapselung; Grundlagen der Objektinteraktion I2.2 Ausnahmen definieren, werfen, behandeln, traversieren I2.3 für Text, Zeit, Datum, Zufall, große Zahlen, Währungen, Logging etc. I2.4 für Code-Generierung, Code-Navigation, Diagrammerzeugung, Dekompilierung, Code-Dokumentation, Code-Verbesserung, VCS-Integration etc.	**3. Semester:** Q2.1 Watches, Evaluierung von Ausdrücken, Live-Änderung von Variablenwerten etc. Q2.2 Nutzung von Codedokumentation; Strategien und Werkzeuge zur Erstellung von Codedokumentation Q2.3 Coding-Conventionen, Style-Guides	**3. Semester:** E2.1 VCS-Setup und Konfiguration; VCS-Werkzeuge und IDE-Integration; grundlegende VCS-Funktionen aus den Bereichen Snapshots (Commits), Inspection, Comparison
	4. Semester: P2.2 Zweck, Herausforderungen und Methoden (z. B. User-Stories, Epics, Use-Cases) der Anforderungsanalyse; Zweck und Realisierbarkeit von UI-Mockups (insb. im Rahmen der Anforderungsanalyse)	**4. Semester:** I2.5 Grundlagen der Vererbung; Gleichheit, Identität und Vergleich von Objekten; statische Elemente etc. I2.6 Vertreter wesentlicher Algorithmen- und Datenstrukturen-Kategorien; Überblick, Auswahl, Vergleich; Anwendungsfälle I2.7 GUI-Bibliotheken; Event-Handling; GUI-Builder-Tools; Umsetzung von UI-Mockups	**4. Semester:** Q2.4 Laufzeitmessung, Feststellung der Komplexitätsklasse Q2.5 Akzeptanztests (Testszenarien und User Journeys, Testplan, Testfälle, Testdurchführung, Ergebnis-Dokumentation)	**4. Semester:** E2.2 weitere VCS-Funktionen aus den Bereichen Inspection, Comparison, Snapshots, Branching & Merging, Sharing & Update, Patching, Debugging E2.3 Anwendung der Kernkompetenzen aus den Kompetenzbereichen des 2. Jahrganges; Fächerintegration: Media Creation and User Experience (insb. GUI-Design und User Experience)

III. Jahrgang: 5. und 6. Semester				
Kompetenz	**5. Semester:** Schülerinnen und Schüler können P3.1 fortgeschrittene Techniken der objektorientierten Analyse anwenden. (O/H)	**5. Semester:** Schülerinnen und Schüler können I3.1 fortgeschrittene Konzepte der objektorientierten Programmierung anwenden. (H) I3.2 fremde Codes bzw. Softwarekomponenten verwenden und einbinden. (H) I3.3 Daten in Dateien schreiben und aus Dateien lesen. (H)	**5. Semester:** Schülerinnen und Schüler können C3.1 automatische Tests implementieren. (H) C3.2 Codequalität unter Verwendung von Analysewerkzeugen einschätzen und verbessern. (H)	**5. Semester:** Schülerinnen und Schüler können E3.1 Build-Tools und Package Manager zur Abwicklung des Build-Prozesses und zum Management von Abhängigkeiten einsetzen. (H) E3.2 Softwareprojekte planen, organisieren und abwickeln. (H)
	6. Semester: Schülerinnen und Schüler können P3.2 grundlegende Softwarearchitekturkonzepte kennen, anwenden und dokumentieren. (O/H)	**6. Semester:** Schülerinnen und Schüler können I3.4 Daten in Datenbanken speichern, aus Datenbanken abfragen und darstellen. (H) I3.5 grundlegende Konzepte und Methoden der funktionalen Programmierung anwenden. (H) I3.6 einfache Softwarearchitekturen implementieren. (H)	**6. Semester:** Schülerinnen und Schüler können C3.3 Code-Reviews durchführen. (H) C3.4 Qualitätssicherungen für Software durchführen. (H) C3.5 Auslieferungen und den Betrieb von Software durchführen. (H)	**6. Semester:** Schülerinnen und Schüler können E3.3 VCS-Plattformen zur Projektabwicklung anwenden. (H) E3.4 Softwareprojekte mit wirtschaftlichem Anwendungsbezug agil oder hybrid im Team planen und umsetzen. (H)
Lehrstoff	**5. Semester:** P3.1 Planung von Vererbungshierarchien, von komplexeren Klassenbeziehungen und Interaktionsmustern (Komposition, Aggregation, Composition over Inheritance), von polymorphen Strukturen sowie des Einsatzes von Interfaces und abstrakten Klassen; Dokumentation (z. B. mit geeigneten Diagrammen und Auszeichnungssprachen)	**5. Semester:** I3.1 Vererbung (inkl. Vorteile, Nachteile und Alternativen), Umsetzung komplexerer Klassenbeziehungen (Komposition, Aggregation, Delegation), Verwendung von Polymorphie, abstrakten Klassen und Interfaces etc. I3.2 Austauschplattformen für Softwareentwickelnde; Code lesen; Einschätzung von Funktionalität und Qualität sowie verantwortungsvolle Nutzung von fremdem Code; (Drittanbieter-)Bibliotheken; Packages I3.3 Dateizugriff; Dateiformate; Datenformate	**5. Semester:** C3.1 Einsatz von Unit-Tests (inkl. Qualitätskriterien) C3.2 Qualitätskriterien für Code (z. B. in Bezug auf Lesbarkeit, Performance, Robustheit, Code-Smells, Best Practices, Metriken); Werkzeuge zur statische Codeanalyse bzw. zur Prüfung der Codequalität; Refactoring	**5. Semester** E3.1 Management von Abhängigkeiten; Bauen von Software E3.2 lineare bzw. nicht-lineare Vorgehensmodelle; iteratives, inkrementelles, hybrides, agiles Software-Projektmanagement; aktuelle Trends
	6. Semester: P3.2 grundlegende Architekturkonzepte und Beispiele (z. B. einfache Schichtenarchitektur mit Präsentations-, Geschäftslogik- und Datenhaltungsschicht bzw. ORM, entkoppelt mit Interfaces); grundlegende Architektur-Dokumentation (z. B. mit geeigneten Diagrammen und Auszeichnungssprachen)	**6. Semester:** I3.4 Datenbankzugriff (z. B. auf SQL bzw. NoSQL-Datenbanken) I3.5 Konzepte und Methoden (z. B. pure Funktionen, Unveränderlichkeit, Funktionen höherer Ordnung, Currying, Komposition, Rekursion, Filter - Map - Reduce, seiteneffektfreie Programmierung); Sprachkonstrukte (z. B. Lambda-Ausdrücke, funktionale Interfaces, Closures, Delegates, Methodenreferenzen, Optionals, Streams, Records, Pattern Matching); Kombination von Programmiersprachenparadigmen I3.6 Implementierung grundlegender Architekturen (aufbauend auf A3.2)	**6. Semester:** C3.3 Code-Reviews (Zweck und Vorgehensweise); Collective Code Ownership C3.4 QA-Maßnahmen in der Softwareentwicklung (z. B. Softwaretests und Testprotokolle); Produkt-Abnahme (z. B. Abnahmetests und Abnahmeprotokolle) C3.5 grundlegende Methoden und Werkzeuge; Maßnahmen der Produkt-Einführung (z. B. mit Operations-Manuals)	**6. Semester** E3.3 Umsetzung etablierter PM-Vorgehensmodelle (siehe D3.2) mit VCS-Unterstützung z. B. für Kollaboration (Branching-Strategien, Merge-Requests, Code Review-Prozess), Issue Tracking, Arbeitspaketmanagement oder Zeitmanagement E3.4 Anwendung der Kernkompetenzen aus den Kompetenzbereichen des 3. Jahrganges; Fächerintegration: Datenmanagement und KI-Integration (insb. Datenbank-Thematik und Projekt), Business Experience & Future Challenges (Digitales Projektmanagement)

IV. Jahrgang: 7. und 8. Semester				
Kompetenz	**7. Semester:** Schülerinnen und Schüler können P4.1 etablierte Entwurfsmuster und Designprinzipien erklären und deren Einsatz planen. (O/H) P4.2 Softwarearchitekturen nach etablierten Architekturmustern entwerfen. (O/H)	**7. Semester:** Schülerinnen und Schüler können I4.1 Entwurfsmuster und Designprinzipien anwenden. (H) I4.2 Softwarearchitekturen implementieren. (H)	**7. Semester:** Schülerinnen und Schüler können C4.1 testgetriebene Entwicklung anwenden. (H) C4.2 Qualität von Softwarearchitekturen einschätzen und verbessern. (H)	**7. Semester:** Schülerinnen und Schüler können E4.1 Containertechnologien im Softwareentwicklungsprozess nutzen. (H) E4.2 Fortgeschrittene Funktionen von VCS-Plattformen anwenden. (H)
	8. Semester: Schülerinnen und Schüler können P4.3 verteilte und sichere Softwaresysteme planen, entwerfen, kommunizieren. (O/H)	**8. Semester:** Schülerinnen und Schüler können I4.3 die Machbarkeit für Softwaresysteme nachweisen. (H) I4.4 verteilte und sichere Softwaresysteme unter Verwendung aktueller Technologien umsetzen. (H)	**8. Semester:** Schülerinnen und Schüler können C4.3 die Sicherheit von Softwaresystemen einschätzen und verbessern. (O/H) C4.4 Containertechnologien und Plattformen für Test, Deployment und Betrieb von (verteilten) Softwaresystemen auswählen, konfigurieren und anwenden. (H)	**8. Semester:** Schülerinnen und Schüler können E4.3 Softwareprojekte mit wirtschaftlichem Anwendungsbezug planen, umsetzen und präsentieren. (H)
Lehrstoff	**7. Semester:** P4.1 Entwurfsmuster (z. B. Erzeugung, Struktur, Verhalten, ORM, Messaging); Designprinzipien (z. B. SOLID, Clean Code, taktische Muster aus DDD, Design-Prinzipien wie DRY, KISS, YAGNI); Einsatzszenarien; Dokumentation (z. B. mit geeigneten Diagrammen und Auszeichnungssprachen) P4.2 Architekturmuster (z. B. Varianten von Schichtenarchitekturen, Varianten von MVC, Clean Architecture bzw. Ports and Adapters, N-Tier Architektur, CQRS, Event Sourcing, Architekturen nach aktuellen Trends); Architektur-Dokumentation (z. B. mit geeigneten Diagrammen und Auszeichnungssprachen)	**7. Semester:** I4.1 Implementierung von Software unter Verwendung etablierter Entwurfsmuster und Designprinzipien (aufbauend auf A4.1) I4.2 Implementierung von Softwarearchitekturen aufbauend auf A4.2 am Beispiel aktueller Softwaretypen (wie z. B. Web-Apps, Mobile-Apps, Cloud-Apps, Apps mit Cloud-Anbindung, Apps mit KI-Unterstützung)	**7. Semester:** C4.1 TDD-Vorgehensmodelle, Testpyramide bzw. Varianten, weitere Testkategorien (z. B. Integrations-, System-, Regressionstests) etc. C4.2 Qualitätskriterien für Softwarearchitekturen (z. B. Änderbarkeit, Testbarkeit, Erweiterbarkeit, Wartbarkeit, Kriterien basierend auf SOLID / Clean Code / DDD, Metriken, technische Schulden, Anti-Patterns); Werkzeuge zur automatischen Prüfung; Refactoring	**7. Semester** E4.1 Containertechnologien im Entwicklungsprozess E4.2 CI/CD, Paketmanagement, fortgeschrittene PM-Funktionen, Security-Funktionen etc.
	8. Semester: P4.3 weitere Techniken der Anforderungsanalyse (z. B. Domain Storytelling, Event Storming, Collaborative Modelling, Vertiefung bekannter Analysetechniken); Prinzipien (z. B. strategische Muster aus DDD, Prinzipien für 12-Factor-Apps), Herausforderungen (z. B. fachlicher Schnitt, Kommunikation, Asynchronität, CAP, eventuale Konsistenz, Ausfallsicherheit, Skalierung, Latenz, Sicherheit, Deployment und Infrastruktur) verteilter Systeme (z. B. Request-Response-Systeme, Socket-basierte Systeme, eventgetriebene Systeme, Microservices, Mischformen, aktuelle Trends); Sicherheitskonzepte (z. B. Authentifizierung, Autorisierung, IAM, RBAC, kryptografische Verfahren, Hashing, Geheimnisverwaltung, bekannte Schachstellen und Gegenmaßnahmen); Dokumentation	**8. Semester:** I4.3 Proof of Concept, technisches Prototyping, rapid Prototyping etc. I4.4 Anwendung von Prinzipien, Techniken, Konzepten etc. aufbauend auf A4.2 und A4.3; Verwendung aktueller Bibliotheken, Frameworks, Plattformen;	**8. Semester:** C4.3 (automatische) Prüfung der Codesicherheit; Refactoring C4.4 Containertechnologien und Plattformen für Test, Deployment und Betrieb von Software	**8. Semester** E4.3 Anwendung der Kernkompetenzen aus den Kompetenzbereichen des 4. Jahrganges; Software-Projektmanagement (aufbauend auf D3.2 und D3.3); Fächerintegration: Betriebssysteme und Netzwerkmanagement (insb. Containertechnologien)

V. Jahrgang				
Kompetenz	Schülerinnen und Schüler können P5.1 aktuelle Konzepte, Methoden und Technologien der Softwareentwicklung im Rahmen von praxisorientierten Aufgabenstellungen einsetzen, um Softwaresysteme ingenieurmäßig zu planen und zu entwerfen. (H) P5.2 weitere aktuelle Konzepte, Methoden und Technologien der Softwareentwicklung im Rahmen von Projekten einsetzen, um Softwaresysteme ingenieurmäßig zu planen und zu entwerfen. (H)	Schülerinnen und Schüler können I5.1 aktuelle Konzepte, Methoden und Technologien der Softwareentwicklung im Rahmen von praxisorientierten Aufgabenstellungen anwenden, um Softwaresysteme ingenieurmäßig zu entwickeln. (H) I5.2 weitere aktuelle Konzepte, Methoden und Technologien der Softwareentwicklung im Rahmen von Projekten einsetzen, um Softwaresysteme ingenieurmäßig zu entwickeln. (H)	Schülerinnen und Schüler können C5.1 aktuelle Konzepte, Methoden und Technologien der Softwareentwicklung im Rahmen von praxisorientierten Aufgabenstellungen einsetzen, um die Funktionalität und Qualität von Softwaresystemen sicherzustellen und Softwaresysteme auszuliefern bzw. zu betreiben. (H) C5.2 weitere aktuelle Konzepte, Methoden und Technologien der Softwareentwicklung im Rahmen von Projekten einsetzen, um die Funktionalität und Qualität von Softwaresystemen sicherzustellen und Softwaresysteme auszuliefern bzw. zu betreiben. (H)	Schülerinnen und Schüler können E5.1 aktuelle Konzepte, Methoden und Technologien der Softwareentwicklung im Rahmen von praxisorientierten Aufgabenstellungen einsetzen, um die ingenieurmäßige Entwicklung von Softwaresystemen zu organisieren und zu begleiten. (H) E5.2 weitere aktuelle Konzepte, Methoden und Technologien der Softwareentwicklung im Rahmen von Projekten einsetzen, um die ingenieurmäßige Entwicklung von Softwaresystemen zu organisieren und zu begleiten. (H)
Lehrstoff	P5.1 aktuelle Trends, Konzepte, Technologien und Methoden für Planung und Entwurf von Software; praxisorientierte Aufgabenstellungen P5.2 aktuelle Trends, Konzepte, Technologien und Methoden für Planung und Entwurf von Software; Projekte (insb. mit Wirtschaftsbezug)	I5.1 aktuelle Trends, Konzepte, Technologien und Methoden zur Implementierung von Software (z. B. aus den Bereichen KI, Cyber Security, Cross-Plattform-Development, IoT, Microcontroller-Programmierung, AR/VR, dezentrale Systeme, verteilte Systeme bzw. Cloud-native, Softwarearchitekturen, Frameworks); praxisorientierte Aufgabenstellungen I5.2 aktuelle Trends, Konzepte, Technologien und Methoden zur Implementierung von Software; Projekte (insb. mit Wirtschaftsbezug)	C5.1 aktuelle Trends, Konzepte, Technologien und Methoden für Test, Deployment und Betrieb von Software; Praxisorientierte Aufgabenstellungen C5.2 aktuelle Trends, Konzepte, Technologien und Methoden für Test, Deployment und Betrieb von Software; Projekte (insb. mit Wirtschaftsbezug)	E5.1 aktuelle Trends, Konzepte, Technologien und Methoden zur Organisation und Unterstützung von Softwareentwicklungsprozessen; Praxisorientierte Aufgabenstellungen E5.2 aktuelle Trends, Konzepte, Technologien und Methoden zur Organisation und Unterstützung von Softwareentwicklungsprozessen; Projekte (insb. mit Wirtschaftsbezug)
Legende: Kompetenzen = Bildungs- und Lehraufgaben; O … Orientierungskompetenz; H … Handlungskompetenz				