Modulhandbuch

für den Studiengang Bachelor Software Engineering

Fakultät für Informatik und Wirtschaftsinformatik

gültig für das Wintersemester 2024 und Sommersemester 2025

Inhaltsverzeichnis

Semester 1

Algebra (1510600)

Algebra

Art des Moduls Pflichtmodul	**Sprache** Deutsch	**SWS** 4	**ECTS** 5
Häufigkeit Jedes Wintersemester	**Dauer** 1 Semester	**Studiensemester** 1	**Lehr- und Lernformen** Seminaristischer Unterricht, Übung
Modulverantwortung	Prof. Dr. Andreas Keller		
Dozierende	Prof. Dr. Andreas Keller		
Verwendbarkeit			
Aufwand	*Gesamt* 150	*Präsenzzeit* 30	*Selbststudium* 120
Voraussetzungen	*nach SPO*: keine *empfohlen*: Beherrschung der Schulmathematik		
Prüfung	*Art der Prüfung*: Schriftliche Prüfung *Art der Note*: Differenzierte Note		
Lernergebnisse	Die Studierenden kennen grundlegende mathematische Konzepte und Begriffe, die in Informatik und Technik Anwendung finden. Die Studierenden verstehen die Bedeutung mathematischer Methoden und deren Rolle in der Entwicklung von Informatikanwendungen. Die Studierenden wenden mathematische Techniken an, um praktische Probleme in der Informatik und Technik zu lösen. Die Studierenden analysieren mathematische Probleme, um geeignete Lösungsstrategien zu identifizieren und zu entwickeln. Die Studierenden bewerten verschiedene Lösungsansätze und deren Effektivität im Kontext spezifischer mathematischer Herausforderungen. Die Studierenden erstellen komplexe mathematische Modelle, die in realen Anwendungen der Informatik und Technik genutzt werden können.		
Modulinhalte	Allgemeine Grundlagen: • Körper der reelle Zahlen • Prinzip der vollständige Induktion • Einführung in den Körper der komplexe Zahlen Lineare Algebra: • Vektorräume (lineare Unabhängigkeit, Basis und Dimension) • Matrizen (Rechnen mit Matrizen, Spur und Determinante) • Lineare Gleichungssysteme • Lineare Abbildungen • Eigenwerte und Eigenvektoren		
Literatur	Gramlich, Günter: Lineare Algebra – Eine Einführung; Fachbuchverlag Leipzig im Carl Hanser Verlag 2019 Hartmann, Peter: Mathematik für Informatiker; Vieweg + Teubner, Wiesbaden 2018 Papula, Lothar: Mathematik für Ingenieure und Naturwissenschaftler 1 und 2; Vieweg + Teubner; Wiesbaden 2018 Schubert, Matthias: Mathematik für Informatiker; Vieweg + Teubner, Wiesbaden 2021 Strang, Gilbert: Lineare Algebra; Springer-Verlag, Berlin/Heidelberg/New York 2016		

Datenbanken (1510400)

Databases

Art des Moduls Pflichtmodul	**Sprache** Deutsch	**SWS** 4	**ECTS** 5
Häufigkeit Jedes Wintersemester	**Dauer** 1 Semester	**Studiensemester** 1	**Lehr- und Lernformen** Seminaristischer Unterricht, Übung
Modulverantwortung	Michael Rott		
Dozierende	Michael Rott		
Verwendbarkeit			
Aufwand	*Gesamt* 150	*Präsenzzeit* 15	*Selbststudium* 135
Voraussetzungen	*nach SPO*: keine *empfohlen*: keine		
Prüfung	*Art der Prüfung*: Schriftliche Prüfung *Art der Note*: Differenzierte Note		
Lernergebnisse	• Die Studierenden können grundlegende Konzepte der Datenpersistenz und die Unterschiede zwischen persistenten und nicht-persistenten Daten erläutern. • Sie können die zentralen Begriffe der relationalen Datenbanken, wie Relation, Primärschlüssel, Fremdschlüssel und Normalisierung, definieren. • Die Studierenden verstehen die Relationale Algebra und können einfache Operationen darauf anwenden. • Sie können den Zusammenhang zwischen konzeptioneller, logischer und physischer Datenmodellierung erklären und deren Bedeutung für die Datenbankentwicklung begründen. • Die Studierenden sind in der Lage, Entity-Relationship-Modelle (ERM) für gegebene Anwendungsfälle zu erstellen und diese in relationale Schemata zu überführen. • Sie können SQL-Abfragen zur Datenmanipulation (DML) und Schema-Definition (DDL) formulieren und ausführen. • Die Studierenden können bestehende Datenbankschemata analysieren und hinsichtlich Redundanz, Konsistenz und Normalformen bewerten. • Sie sind in der Lage, fachliche Informationsbedarfe zu analysieren und daraus geeignete Datenstrukturen und Abfragen abzuleiten.		
Modulinhalte	Das Modul vermittelt die grundlegenden Konzepte und Techniken der Datenbankentwicklung. Es werden das relationale Datenmodell und die Relationen-Algebra als theoretische Grundlagen vorgestellt. Ein Schwerpunkt liegt auf der Datenbankmodellierung, insbesondere der Erstellung von Entity-Relationship-Modellen (ER-Modelle) und deren Überführung in relationale Schemata unter Berücksichtigung von Normalformen. Die Studierenden erwerben praktische Fähigkeiten in SQL, einschließlich der Datenmanipulation, Datenabfrage sowie der Definition von Schemata und der Transaktionsverwaltung. Praktische Übungen und Projekte ermöglichen den Studierenden, ihre Kenntnisse in der Datenbankentwicklung und -administration anzuwenden und eigenständige Datenbanklösungen zu entwerfen und zu implementieren.		
Literatur	Kemper, A., & Eickler, A. (2015). Datenbanksysteme – Eine Einführung (10. Auflage). München: Oldenbourg Verlag Elmasri, R., & Navathe, S. B. (2015). Grundlagen von Datenbanksystemen (7. Auflage). München: Pearson Studium Saake, G., Sattler, K.-U., & Heuer, A. (2011). Datenbanken – Konzepte und Sprachen (3. Auflage). München: Pearson Studium		

	Garcia-Molina, H., Ullman, J. D., & Widom, J. (2013). Database Systems: The Complete Book (2nd ed.). Upper Saddle River, NJ: Pearson

Grundlagen Informatik (1510500)

Introduction to Computer Science

Art des Moduls Pflichtmodul	**Sprache** Deutsch	**SWS** 4	**ECTS** 5
Häufigkeit Jedes Wintersemester	**Dauer** 1 Semester	**Studiensemester** 1	**Lehr- und Lernformen** Seminaristischer Unterricht, Übung
Modulverantwortung	Prof. Dr. Peter Braun		
Dozierende	Prof. Dr. Peter Braun		
Verwendbarkeit	Bachelor Wirtschaftsinformatik		
Aufwand	*Gesamt* 150	*Präsenzzeit* 15	*Selbststudium* 135
Voraussetzungen	*nach SPO*: keine *empfohlen*: keine		
Prüfung	*Art der Prüfung*: Schriftliche Prüfung *Art der Note*: Differenzierte Note		
Lernergebnisse	Nach erfolgreicher Teilnahme an diesem Modul erwerben die Studierenden folgende Kompetenzen: Die Studierenden erinnern grundlegende Begriffe der Informationsverarbeitung und können den Informationsgehalt von Nachrichten messen. Die Studierenden verstehen Codierungen von Daten und grundlegende Methoden zur Modellbildung innerhalb der Informatik. Die Studierenden wenden Verfahren zur Beschreibung von Datenstrukturen an und können einfache Algorithmen zur Datenverarbeitung implementieren. Die Studierenden analysieren einfache dynamische Systeme und beschreiben diese mit Zustandsdiagrammen. Die Studierenden bewerten ethische Fragestellungen im IT-Umfeld anhand konkreter Beispiele. Die Studierenden erstellen reguläre Grammatiken und lösen das Wortproblem mit Hilfe von endlichen Automaten. Die Studierenden erstellen Programme in Python zur Automatisierung von einfachen Aufgaben der Datenverarbeitung.		
Modulinhalte	Das Modul vermittelt die Grundlagen der Informatik für Studierende außerhalb der Kern-Informatik. Information, Informationsgehalt, Informationscodierung, Darstellung von Zahlen und Zeichen, Codierung von Text, Datumsangaben, Farbinformationen Binärarithmetik, Boole'sche Algebra und Logikgatter Modelle und Modellbildung als grundlegendes Prinzip in der Informatik, Abstraktion, Reduktion, Dekomposition, Aggregation Beschreibung von Datenstrukturen mit der erweiterten Backus-Naur-Form Modellierung dynamischer Systeme und ihre Beschreibung mit endlichen Automaten und Zustandsdiagrammen Formale Sprachen, reguläre Grammatiken und das Wortproblem Weitere Automatenmodelle: Moore und Mealy Automaten Der Begriff des Algorithmus, Berechenbarkeit, Halteproblem, Funktionsweise und Programmierung von Turing-Maschinen Grundlegende Algorithmen zum Suchen und Sortieren Geschichte der Hardwareentwicklung Aufbau und prinzipielle Arbeitsweise eines Computers und Mikroprozessors, Von-Neumann Architektur, Moore'sches Gesetz Aufbau und Funktionsweise des Internet und World Wide Web		

	Einführung in die Sprachen HTML und Markdown Aufbau von verteilten Systemen, Client-Server, Peer-to-Peer, Blockchain, Git Kurze Einführung in die Programmierung mit Python Geschichte der Künstlichen Intelligenz, Verfahren des maschinellen Lernens, Regression, Funktionsweise von neuronalen Netzen Datenschutz und Ethik in der Informatik
Literatur	Gumm, Heinz-Peter; Sommer, Manfred: Einführung in die Informatik, 10. Auflage, Oldenbourg, 2013. Ernst, Schmidt, Beneken: Grundkurs Informatik: Grundlagen und Konzepte für die erfolgreiche IT-Praxis, Springer Verlag, 2020

IT-Projektmanagement und BWL (1510100)

IT Project Management and Business Administration

Art des Moduls Pflichtmodul	**Sprache** Deutsch	**SWS** 4	**ECTS** 5
Häufigkeit Jedes Wintersemester	**Dauer** 1 Semester	**Studiensemester** 1	**Lehr- und Lernformen** Seminaristischer Unterricht, Übung
Modulverantwortung	Prof. Dr. Eva Wedlich		
Dozierende	Prof. Dr. Eva Wedlich		
Verwendbarkeit			
Aufwand	*Gesamt* 150	*Präsenzzeit* 15	*Selbststudium* 135
Voraussetzungen	*nach SPO*: keine *empfohlen*: keine		
Prüfung	*Art der Prüfung*: Schriftliche Prüfung *Art der Note*: Differenzierte Note		
Lernergebnisse	Die Studierenden erlernen Projektmanagement-Kompetenzen, insbesondere die notwendigen Kenntnisse für Projektleiter/-innen. Hierzu werden Projektmanagement-Methoden, -Prozesse und -Hilfsmittel behandelt. Die Studierenden lernen Grundbegriffe der Betriebswirtschaftslehre kennen und können diese wiederholen. Im Bereich der Betriebswirtschaftslehre können insbesondere die konstitutive Entscheidungen eines Unternehmens nachvollzogen und die betriebswirtschaftliche Funktionen analysiert werden. Die Studierenden können ökonomische Zusammenhänge nachvollziehen und konstruieren. Die Studierenden sind in der Lage wirtschaftswissenschaftliche Texte (u.a. auch aus Wirtschaftszeitungen) zu verstehen und richtig zu interpretieren.		
Modulinhalte	• Einführung Projekt und Projektmanagement • Projektorganisation • Projektplanungsprozess • Projektkalkulation • Projektsteuerung und –überwachung • Projektabschluss Grundbegriffe der Betriebswirtschaftslehre: • Der Betrieb • Die betriebswirtschaftlichen Produktionsfaktoren • Betriebswirtschaftliche Ziele • Betriebswirtschaftliche Kennzahlen Konstitutive Entscheidungen eines Betriebes: • Standortwahl: • Rechtsformen: Betriebswirtschaftliche Funktionen: • Beschaffung und Einkauf • Marketing und Vertrieb		
Literatur	Vahs, D.; Schäfer-Kunz, J.: Einführung in die Betriebswirtschaftslehre; 16. Auflage; Schäffer-Poeschel, Stuttgart, 2021 Wöhe, G.: Einführung in die allgemeine Betriebswirtschaftslehre; 28. Auflage; Vahlen; München, 2024 Hanke, D.: Die 10 wichtigsten Methoden im Projektmanagement: Der schnelle, verständliche und bewährte Einstieg ins klassische Projektmanagement, 2022		

	Timinger, H.: Modernes Projektmanagement: Mit traditionellem, agilem und hybridem Vorgehen zum Erfolg, Wiley, 2024

Introduction to Software Engineering (1510300)

Introduction to Software Engineering

Art des Moduls Pflichtmodul	**Sprache** Englisch	**SWS** 4	**ECTS** 5
Häufigkeit Jedes Wintersemester	**Dauer** 1 Semester	**Studiensemester** 1	**Lehr- und Lernformen** Seminaristischer Unterricht, Übung
Modulverantwortung	Prof. Dr.-Ing. Anne Heß		
Dozierende	Prof. Dr.-Ing. Anne Heß		
Verwendbarkeit			
Aufwand	*Gesamt* 150	*Präsenzzeit* 15	*Selbststudium* 135
Voraussetzungen	*nach SPO*: keine *empfohlen*: keine		
Prüfung	*Art der Prüfung*: Schriftliche Prüfung *Art der Note*: Differenzierte Note		
Lernergebnisse	Upon successful completion of this module, students will be able to: • Explain the principles of software engineering and assess their relevance • Compare and justify the application of different software development processes • Systematically collect, model, and specify software requirements • Use UML diagrams to structure and analyze software systems • Evaluate software projects in terms of feasibility, cost, and quality • Understand fundamental quality assurance methods • Consider data protection, ethics, and security in software projects		
Modulinhalte	Software engineering covers all phases of software development, from the initial idea to the tested and delivered system. This module introduces the fundamental principles and concepts of software engineering, focusing on systematic development processes, requirements engineering, and basic modeling techniques. It provides the foundation for advanced modules on Analysis and Design and Software Quality. Fundamentals of Software Engineering • Objectives and principles of software engineering • Characteristics and challenges of software projects • Overview of software development activities and processes (agile vs. traditional) Introduction to software (requirements) modeling with UML • Introduction to UML Diagrams: Class Diagrams, Use Case Diagrams, Activity Diagrams, Sequence Diagrams Software Development Processes & Team Collaboration • Roles in software projects (e.g., Product Owner, Developer, Tester) • Fundamentals of project organization and documentation • Cost and benefit estimation in software projects • Agile processes (e.g., Scrum) Quality Aspects and Ethical Considerations • Introduction to software quality (transition to "Software Quality") • Fundamentals of quality assurance (testing, code reviews) • Data protection, ethics, and sustainability in software projects		
Literatur	• Sommerville, Ian: Software Engineering, Pearson, 2018 • Oestereich, Bernd: Analyse und Design mit UML 2.5, Oldenbourg, 2013/2020 • Rupp, Chris: UML glasklar, Hanser, 2012 • McLaughlin: Objektorientierte Analyse und Design von Kopf bis Fuß, O'Reilly, 2017		

Programmieren 1 (1510200)

Programming 1

Art des Moduls Pflichtmodul	**Sprache** Deutsch	**SWS** 4	**ECTS** 5
Häufigkeit Jedes Wintersemester	**Dauer** 1 Semester	**Studiensemester** 1	**Lehr- und Lernformen** Seminaristischer Unterricht, Übung
Modulverantwortung	Prof. Dr. Tristan Wimmer		
Dozierende	Prof. Dr. Tristan Wimmer		
Verwendbarkeit			
Aufwand	*Gesamt* 150	*Präsenzzeit* 15	*Selbststudium* 135
Voraussetzungen	*nach SPO*: keine *empfohlen*: keine		
Prüfung	*Art der Prüfung*: Schriftliche Prüfung *Art der Note*: Differenzierte Note		
Lernergebnisse	Nach erfolgreicher Teilnahme am Modul sind die Studierenden in der Lage: • den Unterschied verschiedener Programmiersprachen zu erklären • den Aufbau von Datenstrukturen zu analysieren • Kontrollstrukturen in Python umzusetzen • Grundlegende Algorithmen zu verstehen und sicher anzuwenden • die Konzepte der objektorientierten Programmierung darzustellen und in Python umzusetzen • das Konzept der Vererbung zu verstehen und umzusetzen		
Modulinhalte	Python zählt zu den wichtigsten Programmiersprachen. In diesem Kurs werden die Studierenden in die Programmierung anhand der Programmiersprache Python eingeführt. Dabei lernen Sie die Klassifizierung von Programmiersprachen kennen. Studierende lernen die wichtigsten Kontrollstrukturen und Datenstrukturen in Python umzusetzen. Des Weiteren machen die Teilnehmenden erste Erfahrungen in der objektorientierten Programmierung.		
Literatur	wird in der Veranstaltung bekannt gegeben		

Semester 2

Algorithmen und Datenstrukturen (1511100)

Algorithms and Datastructures

Art des Moduls Pflichtmodul	**Sprache** Deutsch	**SWS** 4	**ECTS** 5
Häufigkeit Jedes Sommersemester	**Dauer** 1 Semester	**Studiensemester** 2	**Lehr- und Lernformen** Seminaristischer Unterricht, Übung
Modulverantwortung	Prof. Dr.-Ing. Tobias Fertig		
Dozierende	Prof. Dr.-Ing. Tobias Fertig, Prof. Dr.-Ing. Sebastian Biedermann		
Verwendbarkeit			
Aufwand	*Gesamt* 150	*Präsenzzeit* 15	*Selbststudium* 135
Voraussetzungen	*nach SPO*: keine *empfohlen*: keine		
Prüfung	*Art der Prüfung*: Schriftliche Prüfung *Art der Note*: Differenzierte Note		
Lernergebnisse	• Datenstrukturen benennen und bzgl. ihrer Leistungsmerkmale charakterisieren können • Speziellere graph-/baumbasierte Algorithmen benennen, einsetzen und bzgl. ihrer Leistung und Anwendbarkeit charakterisieren können • Für vorgegebene Anwendungsfälle geeignete Datenstrukturen und Algorithmen finden, analysieren und bewerten können • Algorithmen entwickeln und implementieren können • Praktische Erfahrungen beim Einsatz von Algorithmen sammeln Fundierte fachliche Kenntnisse: • Die Studierenden lernen grundlegende Algorithmen und Datenstrukturen kennen Fertigkeit zur Analyse und Strukturierung technischer Problemstellungen: • Die Studierenden lernen, wie sie für vorgegebene Anwendungsfälle geeignete Datenstrukturen und Algorithmen finden und bzgl. ihrer Leistung analysieren Fertigkeit zur Entwicklung und zum Umsetzen von Lösungsstrategien: • Die Studierenden lernen, für praktische Problemstellungen algorithmische Lösungen zu entwickeln und vorhandene Algorithmen einzusetzen Kenntnisse von praxisrelevanten Aufgabenstellungen: • Anhand praktischer Beispiele werden die Einsatzszenarien für verschiedene Algorithmen erarbeitet		
Modulinhalte	Die Veranstaltung behandelt verschiedene komplexere Algorithmen und Datenstrukturen der Informatik in Theorie und praktischer Anwendung. Zur Implementierung der Lösungen können beliebige Programmiersprachen eingesetzt werden. Es werden exemplarisch die folgenden Themenschwerpunkte in Theorie und Praxis behandelt: • Algorithmusbegriff, Datenstrukturen • Stacks, Queues, Listen (mit Optimierungen) • Graphen & verschiedene Algorithmen auf Graphen • Verschiedene Bäume mit jeweiligen Vor- und Nachteilen • Hashmaps und Sondierungsstrategien • Monte-Carlo- und Las-Vegas-Algorithmen • Evolutionäre Algorithmen • Verschlüsselungsalgorithmen und Datenschutz • Dezentrale Software und Blockchain-Datenstrukturen		

Literatur	Saake, Gunter; Sattler, Kai-Uwe: Algorithmen und Datenstrukturen, eine Einführung mit Java; 5. überarb. Aufl.; dpunkt-Verlag; Heidelberg, 2013 Cormen, T., Leiseren, C., Riverest, R., Stein, C.: Algorithmen – Eine Einführung; 3. Aufl.; Oldenburg Verlag, 2010 Weitere Literatur wird in der Veranstaltung bekannt gegeben

Analyse und Design (1510900)

Analysis and Design

Art des Moduls Pflichtmodul	**Sprache** Deutsch	**SWS** 4	**ECTS** 5
Häufigkeit Jedes Sommersemester	**Dauer** 1 Semester	**Studiensemester** 2	**Lehr- und Lernformen** Seminaristischer Unterricht, Übung
Modulverantwortung	Prof. Dr. Isabel John		
Dozierende	Prof. Dr. Isabel John, Prof. Dr.-Ing. Tobias Fertig		
Verwendbarkeit			
Aufwand	*Gesamt* 150	*Präsenzzeit* 15	*Selbststudium* 135
Voraussetzungen	*nach SPO*: keine *empfohlen*: • Introduction to Software Engineering • Programmieren 1		
Prüfung	*Art der Prüfung*: Portfolio *Art der Note*: Differenzierte Note		
Lernergebnisse	Die Studierenden analysieren Anforderungen systematisch, strukturieren sie und dokumentieren diese. Die Studierenden nutzen UML-Diagramme zur Modellierung und Analyse von Software. Die Studierenden konzipieren Softwarearchitekturen basierend auf Entwurfsmustern. Die Studierenden entwerfen Datenbank- und API-Schnittstellen für Anwendungen. Die Studierenden dokumentieren und präsentieren Architekturentscheidungen begründet. Die Studierenden bewerten unterschiedliche Architekturmuster und wenden sie gezielt an. Die Studierenden beherrschen Methoden des objektorientierten Designs im Softwareentwurf.		
Modulinhalte	Dieses Modul baut auf den Grundlagen aus "Introduction to Software Engineering" auf und vermittelt Methoden und Techniken zur systematischen Analyse und dem strukturierten Entwurf von Softwaresystemen. Studierende lernen, komplexe Anforderungen zu analysieren, Softwarearchitekturen zu entwerfen und modellgestützte Entwicklungsansätze anzuwenden. Folgende Themen werden behandelt: Anforderungsanalyse und Spezifikation • Erhebung, Dokumentation und Validierung von Anforderungen • Anwendung von Use Cases, User Stories und Szenarien • Techniken zur Strukturierung und Priorisierung von Anforderungen • Modellierung und Entwurf von Softwarearchitekturen Grundlagen des Design • SOLID-Prinzipien und Clean Code Design • Design Patterns (z. B. Factory, Singleton, Observer) • Datenmodellierung und Schnittstellenentwurf Detaillierte UML-Modellierung (z. B. Klassendiagramme, Komponentendiagramme, Sequenzdiagramme) • Einführung in Architekturmuster wie MVC und Standardarchitekturen (Schichten, Client-Server, Microservices) • Abstraktionsebenen: Vom logischen zum physischen Design Werkzeuge und Dokumentation • Nutzung von CASE-Tools zur Modellierung • Dokumentationstechniken für Architekturentscheidungen		
Literatur	• Sommerville, Ian: Software Engineering, Pearson, 2018 • Oestereich, Bernd: Analyse und Design mit UML 2.5, Oldenbourg, 2013/2020		

	• Rupp, Chris: UML glasklar, Hanser, 2012 • McLaughlin: Objektorientierte Analyse und Design von Kopf bis Fuß, O'Reilly, 2017

Netzwerke (1511000)

Networks

Art des Moduls Pflichtmodul	**Sprache** Englisch	**SWS** 4	**ECTS** 5
Häufigkeit Jedes Sommersemester	**Dauer** 1 Semester	**Studiensemester** 2	**Lehr- und Lernformen** Seminaristischer Unterricht, Übung
Modulverantwortung	Prof. Dr. Rolf Schillinger		
Dozierende	Prof. Dr. Rolf Schillinger		
Verwendbarkeit			
Aufwand	*Gesamt* 150	*Präsenzzeit* 15	*Selbststudium* 135
Voraussetzungen	*nach SPO*: keine *empfohlen*: keine		
Prüfung	*Art der Prüfung*: Schriftliche Prüfung *Art der Note*: Differenzierte Note		
Lernergebnisse	By the end of this module, students will be able to • understand the concepts of layered network architectures and know the different layers and their functionalities • map these concepts onto the Internet Protocol Suite • design a scalable best-practise internetworking architecture for a fictional global company, including suitable choices for link layer protocols, network hardware and interconnections, subnets and routing protocols • build a medium-sized TCP/IP network in a virtual lab • analyze a given network and evaluate its suitability for a given task • troubleshoot network problems common to cloud deployments of web applications		
Modulinhalte	In this module, students will acquire a thorough understanding of current networking technologies, their importance to software engineers, and the numerous security implications that connected systems exhibit. After introducing basic networking concepts, the layered architecture approach as propsed by the ISO/OSI reference model is explained. Since the many different network protocols of the past largely have given way to IP based networking, this course quickly shifts its focus to the internet protocol suite as the foundation on which almost all current (inter)networking is based upon. Each of the four layers is discussed in detail: • Link Layer - Concerned with the physical properties of network connections and the protocols that use these physical connections to build network segments and transfer data between a sender and a receiver within a shared medium. The main focus in this module lies on protocols of the IEEE 802 family for LANs (IEEE 802.3) and wireless networks (IEEE 802.11) with short detours into the realms of 4G and LTE mobile networks. • Network Layer - Allows for sending packets of bits between senders and receivers that are not connected to a shared physical medium, thus extending networks to form potentially global "inter networks", which build the Internet as we currently know it. Main protocols examined in this layer include IPv4/IPv6, ICMP, IPSec and an introduction into routing protocols for interior (IGP) and exterior (BGP) netowrks. • Transport Layer - Providing functionality to transfer data that can span mutliple packets, the transport layer offers connectionsless and connection-oriented transport services. In this module, TCP and UDP are discussed in full detail, with a brief excursion into QUIC. • Application Layer		

	- Software engineers are mainly concerned with the application layer of the IP suite, therefore the widespread HTTP protocol and its encrypted variant HTTPS are covered in detail. In order to maintain operations of computer networks, a host of additional protocols is needed, for example ICMP, DHCP or DNS. These protocols are covered when they become neccessary in the course of examining the different layers of the TCP/IP stack. Networking devices like network interface cards, hubs and switches, bridges and routers, are thoroughly surveyed within their main layers of application as well.
Literatur	Tanenbaum, A., Wetherall, D. & Feamster, N. (2021). Computer Networks. (6th ed.). Pearson International. https://elibrary.pearson.de/book/99.150005/9781292374017

Programmieren 2 (1510800)

Programming 2

Art des Moduls Pflichtmodul	**Sprache** Deutsch	**SWS** 4	**ECTS** 5
Häufigkeit Jedes Sommersemester	**Dauer** 1 Semester	**Studiensemester** 2	**Lehr- und Lernformen** Seminaristischer Unterricht, Übung
Modulverantwortung	Prof. Dr. Peter Braun		
Dozierende	Prof. Dr. Peter Braun		
Verwendbarkeit			
Aufwand	*Gesamt* 150	*Präsenzzeit* 15	*Selbststudium* 135
Voraussetzungen	*nach SPO*: keine *empfohlen*: Programmieren 1		
Prüfung	*Art der Prüfung*: Schriftliche Prüfung *Art der Note*: Differenzierte Note		
Lernergebnisse	Nach erfolgreichem Abschluss des Moduls sind die Studierenden in der Lage: • Grundlegende und fortgeschrittene Kotlin-Sprachkonzepte zu verstehen und anzuwenden. • Objektorientierte Prinzipien in Kotlin zu nutzen und eigene Klassenhierarchien zu entwerfen. • Funktionale Programmierparadigmen in Kotlin anzuwenden, um sauberen und effizienten Code zu schreiben. • Null-Sicherheit und Typsysteme von Kotlin zu verstehen und effektiv einzusetzen. • Testgetriebene Entwicklung (TDD) und Unit-Testing mit Kotlin durchzuführen, um robuste Software zu entwickeln. • Effiziente Fehlerbehandlung unter Nutzung idiomatischer Kotlin-Techniken zu implementieren. • Kotlin-Projekte mit modernen Build-Tools (z. B. Gradle) zu verwalten und Abhängigkeiten sinnvoll einzubinden. • Eine praxisnahe Softwarelösung in Kotlin zu konzipieren, zu implementieren und zu dokumentieren. • Bestehenden Kotlin-Code zu analysieren und auf Qualität, Lesbarkeit und Effizienz zu überprüfen. • Die Unterschiede und Gemeinsamkeiten zwischen Python und Kotlin zu reflektieren und für unterschiedliche Anwendungsfälle abzuwägen.		
Modulinhalte	Das Modul "Programmieren 2" baut auf den im ersten Semester erworbenen Kenntnissen in Python und objektorientierter Programmierung (OOP) auf und führt in die Programmiersprache Kotlin ein. Dabei werden sowohl die Sprachkonzepte von Kotlin als auch moderne Softwareentwicklungsprinzipien vermittelt. Die Schwerpunkte des Moduls sind: • Syntax und Konzepte von Kotlin: Variablen, Kontrollstrukturen, Funktionen und Lambdas • Objektorientierte Programmierung in Kotlin: Klassen, Objekte, Vererbung, Interfaces und Datenklassen • Funktionale Programmieransätze: Higher-Order-Funktionen, Immutability und Collections-APIs • Kotlin-spezifische Konzepte: Null-Safety, Extension Functions, Smart Casts und Coroutines • Testgetriebene Entwicklung (TDD) mit Kotlin und Unit-Tests • Fehlerbehandlung mit Exceptions und idiomatischen Ansätzen in Kotlin • Einführung in moderne Softwareentwicklung mit Kotlin: Nutzung von Build-Tools (Gradle), Abhängigkeitsmanagement, einfache Anwendung von Design Patterns • Projektarbeit: Umsetzung einer praxisnahen Anwendung unter Verwendung von Kotlin		

	• Praktische Übungen und Projekte begleiten die theoretischen Inhalte, sodass die Studierenden eigenständig Kotlin-Programme entwickeln und deren Qualität durch Tests und Code-Reviews verbessern können.
Literatur	• Kofler, Michael. Kotlin: Das Umfassende Handbuch. 1. Auflage. Bonn: Rheinwerk, 2021. • Szwillus, Karl. Kotlin: Einstieg Und Praxis. 1. Auflage. Frechen: mitp, 2020.

Projekt 1 (1510700)

Project 1

Art des Moduls Pflichtmodul	**Sprache** Deutsch	**SWS** 1	**ECTS** 5
Häufigkeit Jedes Sommersemester	**Dauer** 1 Semester	**Studiensemester** 2	**Lehr- und Lernformen** Projekt
Modulverantwortung	Prof. Dr. Peter Braun		
Dozierende	Prof. Dr. Peter Braun, Prof. Dr. Isabel John, Michael Rott, Prof. Dr. Rolf Schillinger, Prof. Dr.-Ing. Tobias Fertig, Prof. Dr. Frank-Michael Schleif, Prof. Dr.-Ing. Sebastian Biedermann, Prof. Dr. Tristan Wimmer, Prof. Dr.-Ing. Anne Heß		
Verwendbarkeit			
Aufwand	*Gesamt* 150	*Präsenzzeit* 15	*Selbststudium* 135
Voraussetzungen	*nach SPO*: keine *empfohlen*: Programmieren 1, IT-Projektmanagement und BWL		
Prüfung	*Art der Prüfung*: Präsentation, Dokumentation *Art der Note*: Differenzierte Note		
Lernergebnisse	Nach dem erfolgreichen Abschluss des Moduls haben die Studierenden die folgenden Kompetenzen: • Die Studierenden erinnern die grundlegenden Konzepte der Softwareentwicklung und Versionskontrolle. • Die Studierenden verstehen die Prinzipien modularer Softwareentwicklung und die Strukturierung von Projekten. • Die Studierenden wenden Git und Versionskontrolle in einem Teamkontext an. • Die Studierenden analysieren nicht-triviale Algorithmen hinsichtlich Laufzeit und Effizienz. • Die Studierenden bewerten verschiedene Implementierungsansätze für eine gegebene Problemstellung. • Die Studierenden erstellen eine kleine, modulare Konsolenanwendung nach vorgegebenen Anforderungen. • Die Studierenden erstellen eine technische Präsentation zur Vorstellung ihrer Lösung.		
Modulinhalte	Die Studierenden entwickeln in einer Gruppe mit drei bis vier Personen eine Konsolenanwendung gemäß vorgegebener Anforderungen. Die Anwendung soll einen nicht-trivialen Algorithmus enthalten (z. B. Such-, Sortier-, Graphen- oder Optimierungsalgorithmen) und mit Dateien arbeiten. Die Software besteht aus mehreren Modulen, jedoch ohne eine umfassende Softwarearchitektur. Zu den zentralen Themen gehören: • Modularisierung des Codes • Effiziente Implementierung eines Algorithmus • Nutzung von Versionskontrolle (z. B. Git) • Grundlegende Code-Dokumentation (README, Funktionskommentare) • Einfache Tests zur Validierung des Algorithmus • Einfache Aufgabenverwaltung und -verteilung im Team		
Literatur			

Stochastik (1511200)

Stochastics

Art des Moduls	**Sprache**	**SWS**	**ECTS**
Pflichtmodul	Deutsch	4	5
Häufigkeit	**Dauer**	**Studiensemester**	**Lehr- und Lernformen**
Jedes Sommersemester	1 Semester	2	Seminaristischer Unterricht, Übung
Modulverantwortung	Prof. Dr. Patrik Stilgenbauer		
Dozierende	Prof. Dr. Patrik Stilgenbauer		
Verwendbarkeit	Bachelor Informatik		
Aufwand	*Gesamt*	*Präsenzzeit*	*Selbststudium*
	150	15	135
Voraussetzungen	*nach SPO*: keine *empfohlen*: Algebra: Kombinatorik, Aussagen- und Mengenalgebra, Matrizenalgebra, Lineare Gleichungssysteme. Programmieren I: Programmierlogik, Entwurf einfacher Algorithmen. Gute Schulkenntnisse aus der Analysis (Differential- und Integralrechnung).		
Prüfung	*Art der Prüfung*: Schriftliche Prüfung *Art der Note*: Differenzierte Note		
Lernergebnisse	Math.-naturwiss. Grundlagen: Die Studierenden lernen die Grundlagen der Mathematik kennen, die für die Stochastik (und insb. auch die Statistik) relevant sind. Fertigkeit zum logischen, analytischen und konzeptionellen Denken: Durch Lösen von Aufgaben aus der Stochastik wird die Fähigkeit zum logischen Denken geschult. Auswahl und sichere Anwendung geeigneter Methoden: An Beispielen und Aufgaben lernen die Studierenden die Auswahl und sichere Anwendung geeigneter Methoden und Verfahren der Stochastik sowie Statistik.		
Modulinhalte	Deskriptive Statistik: Grundbegriffe, Häufigkeitsverteilungen, Lageparameter, Streuungsparameter, Korrelations- und Regressionsrechnung. Wahrscheinlichkeitstheorie: Ergebnismenge, Ereignisse, Wahrscheinlichkeitsbegriff von Kolmogorow, bedingte Wahrscheinlichkeit und Unabhängigkeit, diskrete und stetige Zufallsvariablen, Erwartungswert und Varianz, Binomialverteilung, Hypergeometrische Verteilung, Poissonverteilung, Exponentialverteilung, Normalverteilung, Summen von Zufallsvariablen, zentraler Grenzwertsatz. Schließende Statistik: Punkt- und Intervallschätzungen, Signifikanztests.		
Literatur	Bamberg, G., Baur, F. und Krapp, M.: Statistik, Oldenburg Verlag, München/Wien. Bourier, G.: Beschreibende Statistik, Gabler Verlag, Wiesbaden. Bourier, G.: Wahrscheinlichkeitsrechnung und schließende Statistik, Gabler Verlag, Wiesbaden. Christoph, G. und Hackel, H.: Starthilfe Stochastik, Teubner Verlag, Stuttgart/Leipzig/ Wiesbaden. Dreiseitl, S.: Mathematik für Software Engineering, Springer Vieweg Verlag, Berlin. Greiner, M. und Tinhofer, G.: Stochastik für Studienanfänger der Informatik, Hanser Verlag, München/Wien. Henze, N.: Stochastik für Einsteiger, Vieweg Verlag, Wiesbaden. Teschl, G. und Teschl, S.: Mathematik für Informatiker - Band 2 (Analysis und Stochastik), Springer Vieweg Verlag, Berlin/Heidelberg.		

Semester 3

Backend Systems (1511600)

Backend Systems

Art des Moduls	**Sprache**	**SWS**	**ECTS**
Pflichtmodul	Englisch	4	5
Häufigkeit	**Dauer**	**Studiensemester**	**Lehr- und Lernformen**
Jedes Wintersemester	1 Semester	3	Seminaristischer Unterricht, Übung
Modulverantwortung	Prof. Dr. Peter Braun		
Dozierende	Prof. Dr. Peter Braun		
Verwendbarkeit	Bachelor Informatik		
Aufwand	*Gesamt*	*Präsenzzeit*	*Selbststudium*
	150	15	135
Voraussetzungen	*nach SPO*: keine *empfohlen*: Programmieren 1 und Programmieren 2		
Prüfung	*Art der Prüfung*: Portfolio *Art der Note*: Differenzierte Note		
Lernergebnisse	Understand and describe the fundamental concepts and differences of distributed systems Explain the principles and components of various software architectures for backend systems Implement a backend system using a framework like Spring, applying best practices for configuration and deployment Apply advanced database techniques, including replication and sharding Compare and contrast different protocols for remote procedure calls, such as GraphQL and Google RPC, in terms of their functionality and use cases Apply the basics of the HTTP protocol to design and implement Web APIs, including understanding HTTP methods and status codes Design a RESTful API following the REST architecture principles, incorporating resources, URLs, CRUD operations, hypermedia, caching strategies, and security measures. Configure web servers, load balancers, and public caches to optimize backend system performance. Conduct comprehensive testing of backend systems, including performance testing with tools like JMeter Evaluate the security aspects of network protocols and backend systems, applying best practices for authentication, authorization The topics of the practical examples for the examination are provided by or agreed with the lecturer in the traditional degree programme. In the BIN dual study programme, a task from the company is worked on in consultation with the lecturer. This ensures practical relevance and feedback from the company.		
Modulinhalte	Introduction to distributed systems, client-server, and peer-to-peer systems. Software architectures for backend systems (3-tier, hexagonal, monolithic vs. micro-service, event-driven) Frameworks to implement backend systems (e.g., Spring) Advanced database techniques, scalability, replication, sharding, ORM tools, query caching, CAP theorem Protocols for remote procedure calls, for example, GraphQL and Google RPC Basics of the HTTP protocol and application in the form of Web APIs A comprehensive introduction to the REST architecture principle: resources, URLs, CRUD, hypermedia, caching, and security Configuration of Web servers (Apache), load balancer, and public caches (nginx) Testing of backend systems, performance testing using JMeter, monitoring and logging Security aspects of network protocol and backend systems		

	Introduction to Cloud Computing, Cloud-Service models, and cloud platforms (e.g., Google or AWS) Introduction to Kubernetes Introduction to DevOps and CI/CD pipelines for backend systems
Literatur	[1] D. J. Harkness, Apache Essentials: Install, Configure, Maintain. Berkeley, CA: Apress, 2022. doi: 10.1007/978-1-4842-8324-0. [2] Coulouris, J. Dollimore, und T. Kindberg, Distributed Systems: Concepts and Design (4th Edition) (International Computer Science). Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005. [3] C. Surianarayanan und P. R. Chelliah, Essentials of Cloud Computing: A Holistic, Cloud-Native Perspective. in Texts in Computer Science. Cham: Springer International Publishing, 2023. doi: 10.1007/978-3-031-32044-6. [4] S. Pandya und R. Guha Thakurta, Introduction to Infrastructure as Code: A Brief Guide to the Future of DevOps. Berkeley, CA: Apress, 2022. doi: 10.1007/978-1-4842-8689-0. [5] P. Martin, Kubernetes: preparing for the CKA and CKAD certifications. New York, NY: Apress, 2021. [6] D. DeJonghe, NGINX cookbook: advanced recipes for operations, First edition. Sebastopol, CA: O'Reilly Media, 2017. [7] N. Biswas, Practical GraphQL: Learning Full-Stack GraphQL Development with Projects. Berkeley, CA: Apress, 2023. doi: 10.1007/978-1-4842-9621-9. [8] B. Parasuraman, Practical Spring Cloud Function: Developing Cloud-Native Functions for Multi-Cloud and Hybrid-Cloud Environments. Berkeley, CA: Apress, 2023. doi: 10.1007/978-1-4842-8913-6. [9] S. Matam und J. Jain, Pro Apache JMeter. Berkeley, CA: Apress, 2017. doi: 10.1007/978-1-4842-2961-3. [10] J. Webber, S. Parastatidis, und I. Robinson, REST in practice: hypermedia and systems architecture, 1. ed. in Theory in practice. Beijing Köln: O'Reilly, 2010. [11] L. Richardson und M. Amundsen, RESTful Web APIs, First edition, Second release. Beijing Cambridge Farnham Köln Sebastopol Tokyo: O'Reilly, 2015. [12] I. Dominte, Web API Development for the Absolute Beginner: A Step-by-step Approach to Learning the Fundamentals of Web API Development with .NET 7. Berkeley, CA: Apress, 2023. doi: 10.1007/978-1-4842-9348-5.

Data Science (1511700)

Data Science

Art des Moduls	**Sprache**	**SWS**	**ECTS**
Pflichtmodul	Englisch	4	5
Häufigkeit	**Dauer**	**Studiensemester**	**Lehr- und Lernformen**
Jedes Wintersemester	1 Semester	3	Seminaristischer Unterricht, Übung
Modulverantwortung	Prof. Dr. Frank-Michael Schleif		
Dozierende	Prof. Dr. Frank-Michael Schleif		
Verwendbarkeit			
Aufwand	*Gesamt*	*Präsenzzeit*	*Selbststudium*
	150	15	135
Voraussetzungen	*nach SPO*: Datenbanken I, Programmieren I, Software Engineering I, Programmieren II *empfohlen*: keine		
Prüfung	*Art der Prüfung*: Schriftliche Prüfung, Portfolio *Art der Note*: Differenzierte Note		
Lernergebnisse	Upon successful completion of this module, students will be able to: Knowledge and Understanding: • Explain fundamental **data management principles** and their role in modern software engineering. • Describe different **data models and formats**, including relational, hierarchical, and graph-based representations. • Understand **semi-structured data technologies**, including **XML, JSON, XPath, and XSLT**. • Discuss **data integration techniques**, including **ETL pipelines, data warehousing, and NoSQL databases**. • Explain the basics of **graph databases**, including **modeling and querying with Cypher**. • Demonstrate **data literacy skills**, including ethical considerations, privacy regulations, and responsible data handling. Skills and Competences: • Design and implement **data models** for structured and semi-structured data. • Develop and execute **queries** on relational, JSON, and graph databases. • Construct **ETL workflows** for data integration and preprocessing. • Use **data warehousing and OLAP techniques** to analyze and aggregate data for decision support. • Apply **graph-based data analysis** for complex networked data structures. • Critically assess the quality, security, and privacy aspects of data storage and processing.		
Modulinhalte	Data is a fundamental asset in modern software engineering, driving decision-making, business intelligence, and innovative applications. This module introduces students to essential **data management, modeling, and data science** concepts, establishing a strong foundation for advanced topics such as **machine learning** in later semesters. The course covers the complete **data lifecycle**, from data representation and integration to analytical processing and visualization. Students will gain **practical experience** with various data models, including **relational, semi-structured (XML/JSON), NoSQL, and graph databases**, while also exploring **ETL (Extract, Transform, Load) processes and data warehousing. An introduction to **data literacy** ensures that students develop a critical understanding of data interpretation, privacy concerns, and ethical implications in data-driven applications.		

	By integrating theoretical foundations with **hands-on exercises**, the course prepares students for real-world data challenges, emphasizing both **technical and problem-solving competencies** in software-driven data management. The module is structured into the following core topics: • *1. Introduction to Data Science and Data Literacy** • The role of data in software engineering • Basics of data-driven decision-making • Ethical and legal considerations in data handling (GDPR, data privacy) • Data quality and bias in data science applications • *2. Semi-Structured Data: XML and JSON Technologies** • Fundamentals of **XML and JSON** as data representation formats • Schema definitions: **DTD and XML Schema** • Querying **XML and JSON documents** (XPath, XSLT, JSONPath) • Use cases and industry applications of XML/JSON • *3. Data Warehousing and Integration** • **Multidimensional data modeling** (star schema, snowflake schema) • **ETL processes**: Data extraction, transformation, and loading • **Data integration** from relational databases, web services, and APIs (JDBC/ODBC) • **OLAP (Online Analytical Processing)** and its role in data analytics • Introduction to **NoSQL and Big Data technologies** • *4. Graph Databases and Networked Data** • Fundamentals of **graph theory and graph data management** • **Graph database models** and storage structures • Querying with **Cypher** and data analysis techniques • Applications of graph-based data science (e.g. social networks, recommendation systems) • *5. Practical Applications and Case Studies** • Designing and implementing **data-driven solutions** for software engineering challenges • **Performance considerations** in large-scale data processing • Security, privacy, and compliance in data management
Literatur	Skiena, S.S.; The Data Science Design Manual, Springer, 2017 Robinson, I; Graph Databases 2nd Ed.; O'Reilly Media; 2015 Friesen, Jeff; Java XML and JSON; 2019 Brian Knight, Professional Microsoft SQL Server 2014 Integration Services (Wrox Programmer to Programmer), Wrox, 2014 Trevor Hastie, The Elements of Statistical Learning, Springer, 2009 Ralph Kimball, Margy Ross, Warren Thornthwaite, Joy Mundy, Bob Becker: The Data Warehouse Lifecycle Toolkit, 2nd Edition, Wiley 2008 (further literature maybe provided in the lecture)

Professional Skills (1511900)

Professional Skills

Art des Moduls Pflichtmodul	**Sprache** Deutsch	**SWS** 4	**ECTS** 5
Häufigkeit Jedes Wintersemester	**Dauer** 1 Semester	**Studiensemester** 3	**Lehr- und Lernformen** Seminaristischer Unterricht
Modulverantwortung	Prof. Dr. Isabel John		
Dozierende	Prof. Dr. Peter Braun, Prof. Dr. Isabel John, Prof. Dr.-Ing. Tobias Fertig, Prof. Dr. Frank-Michael Schleif, Prof. Dr.-Ing. Anne Heß		
Verwendbarkeit	Bachelor Informatik		
Aufwand	*Gesamt* 150	*Präsenzzeit* 15	*Selbststudium* 135
Voraussetzungen	*nach SPO*: keine *empfohlen*: keine		
Prüfung	*Art der Prüfung*: Portfolio *Art der Note*: Differenzierte Note		
Lernergebnisse	• Die Studierenden wenden Methoden zur effektiven Planung und Strukturierung ihrer Arbeitsprozesse an • Die Studierenden benennen die Grundprinzipien des wissenschaftlichen Arbeitens in der Informatik • Die Studierenden führen eine Literaturrecherche durch und organisieren die Ergebnisse • Die Studierenden beschreiben detailliert die Grundlagen zur Gestaltung effektiver wissenschaftlicher Kommunikation in Form von Texten, Präsentationen, Poster, Videos • Die Studierenden erstellen wissenschaftlich-orientierte Texte, Präsentationen, Poster, Videos auf der Basis wissenschaftlicher Standards • Die Studierenden erlernen relevante Zielsetzungen, Fähigkeiten und Best Practices zur Planung, Durchführung und Nachbereitung verschiedener Befragungstechniken (wie Interviews, Umfragen) • Die Studierenden wenden Methoden zur effektiven Kommunikation in Teams an • Die Studierenden wenden Methoden der Gesprächsführung zur Ermittlung von Anforderungen an		
Modulinhalte	In diesem Modul werden den Studierenden theoretische und praktische Kenntnisse und Fähigkeiten vermittelt, die in einem professionellen Arbeitsumfeld in verschiedensten Bereichen Anwendung finden. Dabei erlernen und erproben die Studierenden eine Reihe von Methoden, Techniken, und Tools, die in verschiedene Schwerpunktthemen eingeordnet sind. Dazu gehören • Lern- und Arbeitstechniken • Wissenschaftliches Arbeiten • Zielgruppenorientierte fachliche Kommunikation sowie • Arbeiten in (internationalen) Teams		
Literatur	Wird in Vorlesung bekanntgegeben		

Projekt 2 (1511300)

Project 2

Art des Moduls Pflichtmodul	**Sprache** Deutsch	**SWS** 1	**ECTS** 5
Häufigkeit Jedes Wintersemester	**Dauer** 1 Semester	**Studiensemester** 3	**Lehr- und Lernformen** Projekt
Modulverantwortung	Prof. Dr. Peter Braun		
Dozierende	Prof. Dr. Peter Braun, Prof. Dr. Isabel John, Michael Rott, Prof. Dr. Rolf Schillinger, Prof. Dr.-Ing. Tobias Fertig, Prof. Dr. Frank-Michael Schleif, Prof. Dr.-Ing. Sebastian Biedermann, Prof. Dr. Tristan Wimmer, Prof. Dr.-Ing. Anne Heß		
Verwendbarkeit			
Aufwand	*Gesamt* 150	*Präsenzzeit* 15	*Selbststudium* 135
Voraussetzungen	*nach SPO*: keine *empfohlen*: keine		
Prüfung	*Art der Prüfung*: Dokumentation, Präsentation *Art der Note*: Differenzierte Note		
Lernergebnisse	Nach erfolgreichem Abschluss des Moduls besitzen die Studierenden die folgenden Kompetenzen: Die Studierenden verstehen grundlegende Architekturprinzipien wie das Schichtenmodell und die Modularisierung. Die Studierenden wenden das Schichtenmodell und Modularisierungskonzepte bei der Entwicklung einer verteilten Anwendung mit Datenbankanbindung an. Die Studierenden wenden agile Methoden an, um Aufgaben in Sprints zu planen und umzusetzen. Die Studierenden analysieren Code-Änderungen im Rahmen von Pull Requests und führen Code Reviews im Team durch. Die Studierenden erstellen eine strukturierte Software-Dokumentation für ihre Anwendung. Die Studierenden erstellen eine technische Projektpräsentation zur Vorstellung ihrer Ergebnisse.		
Modulinhalte	In Gruppen von vier bis fünf Personen entwickeln die Studierenden eine Anwendung mit dem Fokus auf das Backend und die Datenbank. Sie wenden Elemente des agilen Projektmanagement an. Die Studierenden erhalten ein Thema und ermitteln mit der betreuenden Personen die Anforderungen. Eine grafische Nutzeroberfläche ist nicht notwendig bzw. sollte sehr einfach gehalten werden. Die Software muss eine erkennbare Software Architektur (z.B. Schichten, Pipeline, Hexagonal) aufweisen. Die Qualitätssicherung erfolgt durch Unit-Tests. Die Studierenden arbeiten mit: • Datenbank-Anbindung mit Schema-Entwurf (SQL) • Erweiterten Software-Entwurfsmethoden (z. B. UML, Architektur-Patterns) • Versionskontrolle mit Git (Branching, Pull Requests, Code Reviews) • Sprint-basierten Entwicklungsprozessen (z. B. Scrum, Kanban) • Die Dokumentation umfasst Architekturdiagramme, User Stories und eine automatische API-Dokumentation. • Maßnahmen zur Qualitätssicherung durch automatisierte Unit-Tests		
Literatur			

Software Qualität (1511500)

Software Quality

Art des Moduls	**Sprache**	**SWS**	**ECTS**
Pflichtmodul	Deutsch	4	5
Häufigkeit	**Dauer**	**Studiensemester**	**Lehr- und Lernformen**
Jedes Wintersemester	1 Semester	3	Seminaristischer Unterricht, Übung
Modulverantwortung	Prof. Dr.-Ing. Tobias Fertig		
Dozierende	Prof. Dr.-Ing. Tobias Fertig		
Verwendbarkeit			
Aufwand	*Gesamt*	*Präsenzzeit*	*Selbststudium*
	150	15	135
Voraussetzungen	*nach SPO*: keine *empfohlen*: Introduction to Software Engineering Analyse und Design		
Prüfung	*Art der Prüfung*: Portfolio *Art der Note*: Differenzierte Note		
Lernergebnisse	Die Studierenden sollen in der Lage sein, Begrifflichkeiten der Software Qualität zu verstehen und zu erläutern. Zudem lernen die Studierenden Test-Strategien zu entwickeln, Testfälle zu definieren und Testüberdeckungen zu ermitteln. Zusätzlich lernen die Studierenden Testmethoden für funktionale und nicht-funktionale Anforderungen auszuwählen und anzuwenden. Studierende können zudem ein geeignetes Testmanagement etablieren und individuell auf die Anforderungen von Projekten abstimmen.		
Modulinhalte	Dieses Modul leistet eine umfassende Einführung in den Themenkomplex Software Qualität. Dabei werden sowohl technische Grundlagen als auch Prozess- und Management-Themen näher gebracht. Die Inhalte umfassen die folgenden Themen: • Einführung zu Software Tests • Techniken für Tests • Test-Driven Development • Automated Testing Tools • Funktionales Testen, Regressionstests, Performance Tests • Bug Tracking und Reporting • Softwarequalität & Software-Qualitätsmanagement • Code Reviews & Statische Analyse • Change Management in Software • Analytische und Konstruktive QA • Human-Centered Design und Usability Testing		
Literatur	Software-Qualität, Peter Liggesmeyer, 978-3-8274-2056-5		

System-oriented Programming (1511400)

System-oriented Programming

Art des Moduls Pflichtmodul	**Sprache** Englisch	**SWS** 4	**ECTS** 5
Häufigkeit Jedes Wintersemester	**Dauer** 1 Semester	**Studiensemester** 3	**Lehr- und Lernformen** Seminaristischer Unterricht, Übung
Modulverantwortung	Prof. Dr. Peter Braun		
Dozierende	Prof. Dr. Peter Braun		
Verwendbarkeit	Bachelor Informatik		
Aufwand	*Gesamt* 150	*Präsenzzeit* 15	*Selbststudium* 135
Voraussetzungen	*nach SPO*: keine *empfohlen*: Programmieren 1 and Programmieren 2		
Prüfung	*Art der Prüfung*: Portfolio *Art der Note*: Differenzierte Note		
Lernergebnisse	• Understand the definition and meaning of system-oriented programming. • Demonstrate proficiency in using the command line of an operating system, including advanced tools like sed, awk, sort, and jq for data processing. • Develop shell scripts using Bash to automate tasks and manage system operations. • Apply remote computer access techniques using ssh for secure and efficient work. • Utilize AI assistance systems such as GitHub Copilot and ChatGPT to enhance coding efficiency and problem-solving. • Edit and manage text documents effectively using the vim text editor. • Implement version control practices using Git for collaborative software development. • Program in C, covering syntax, data types, pointers, and memory management, and analyze code performance using debugging and profiling tools (gdb, strace, ltrace, gprof). • Evaluate security aspects of system-related programming and implement protection mechanisms to ensure secure code. • Explain the structure of the Linux operating system and the concepts of processes, process management, and scheduling, and design solutions for inter-process communication challenges, including handling race conditions, deadlocks, and implementing synchronization mechanisms like semaphores and Petri nets.		
Modulinhalte	• Definition and meaning of system-oriented programming • Using the command line of an operating system • Shell programming using the example of Bash • Working on remote computers with ssh • Data processing on the command line with sed, awk, sort, jq • Using AI assistance systems (e.g. GitHub Copilot, ChatGPT) • Editing text documents with vim • Using the version control system Git • Introduction to the C programming language (syntax, data types, pointers, memory management) • Build systems make, cmake, bazel • System programming under Linux (system calls, handling files, processes) • Debugging, profiling and performance optimisation (gdb, strace, ltrace, gprof) • Security aspects of system-related programming and protection mechanisms • Structure of the Linux operating system • Processes, process management, scheduling • Inter-process communication, race conditions, deadlocks, semaphores, Petri nets and deadlock detection, philosopher problem, producer-consumer problem		

	• Memory management, memory abstraction, partitioning, fragmentation, free memory management, virtual memory, page exchange algorithms • Input and output, direct memory access, interrupts, hard disks, file systems for hard disks • Backup methods, automation, data integrity • Network communication and implementation of network protocols • Hypervisor technologies, Docker containers, resource management
Literatur	[1] D. J. Barrett, Efficient Linux at the command line: boost your command-line skills, First edition. Sebastopol, CA: O'Reilly, 2022. [2] A. S. Tanenbaum und H. Bos, Modern operating systems, 4. ed. Boston: Prentice Hall, 2015. [3] K. Hitchcock, Linux System Administration for the 2020s: The Modern Sysadmin Leaving Behind the Culture of Build and Maintain. Berkeley, CA: Apress, 2022. doi: 10.1007/978-1-4842-7984-7. [4] M. Kalin, Modern C Up and Running: A Programmer's Guide to Finding Fluency and Bypassing the Quirks. Berkeley, CA: Apress, 2022. doi: 10.1007/978-1-4842-8676-0. [5] K. Hitchcock, The Enterprise Linux Administrator: Journey to a New Linux Career. Berkeley, CA: Apress, 2023. doi: 10.1007/978-1-4842-8801-6. [6] J. Varma, Pro Bash: Learn to Script and Program the GNU/Linux Shell. Berkeley, CA: Apress, 2023. doi: 10.1007/978-1-4842-9588-5. [7] S. M. Palakollu, Practical System Programming with C: Pragmatic Example Applications in Linux and Unix-Based Operating Systems. Berkeley, CA: Apress, 2021. doi: 10.1007/978-1-4842-6321-1.

Semester 4

Allgemeinwissenschaftliches Wahlpflichtmodul (9999999)

General Compulsory Elective

Art des Moduls	**Sprache**	**SWS**	**ECTS**
Pflichtmodul	Deutsch/Englisch	0	5
Häufigkeit	**Dauer**	**Studiensemester**	**Lehr- und Lernformen**
Jedes Semester	1 Semester	4	Seminaristischer Unterricht
Modulverantwortung	Prof. Dr. Jochen Seufert		
Dozierende	Beate Wassermann		
Verwendbarkeit	Bachelor Informatik, Bachelor Wirtschaftsinformatik		
Aufwand	*Gesamt*	*Präsenzzeit*	*Selbststudium*
	150	15	135
Voraussetzungen	*nach SPO*: i. d. R. keine; Ausnahmen werden durch die Fakultät Angewandte Natur- und Geisteswissenschaften festgelegt und bekanntgegeben. *empfohlen*: keine		
Prüfung	*Art der Prüfung*: Schriftliche Prüfung *Art der Note*: Differenzierte Note		
Lernergebnisse	Die fachspezifischen Lernziele sind abhängig von den jeweils ausgewählten AWPF. Die Studierenden • erwerben zudem Wissen und Kompetenzen, die nicht fachspezifisch sind, aber für das angestrebte Berufsziel bedeutsam sein können wie beispielsweise spezielle Kenntnisse bei Fremdsprachen, in naturwissenschaftlichen oder auch in sozialwissenschaftlichen Gebieten • analysieren unterschiedlichste Fragestellungen • ordnen das fachspezifische Wissen in einen interdisziplinären Zusammenhang ein • übertragen das Gelernte auf die aktuelle Ausbildung • haben ihre Schlüsselkompetenzen und ggf. Fremdsprachenkompetenzen erweitert, wodurch die Persönlichkeitsbildung unterstützt wird, auch in interkultureller Hinsicht • sind sich ihrer Verantwortung in persönlicher, gesellschaftlicher und ethischer Hinsicht bewusst.		
Modulinhalte	Auswahl von zwei Allgemeinwissenschaftlichen Wahlpflichtfächern (AWPF) (2 x 2,5 ECTS) bzw. einem AWPF (1 x 5 ECTS) aus dem Fächerangebot der Fakultät Angewandte Natur- und Geisteswissenschaften (FANG). Fächerangebot der FANG aus den Bereichen • Sprachen • Kulturwissenschaften • Naturwissenschaften und Technik • Politik, Recht und Wirtschaft • Pädagogik, Psychologie und Sozialwissenschaften • Soft Skills • Kreativität und Kunst. Ausgeschlossen aus dem Angebotskatalog der FANG sind Veranstaltungen, deren Inhalte bereits Bestandteile oder unmittelbar fachlich verwandt mit Teilen anderer Module des Studiengangs sind. Die entsprechenden Veranstaltungen sind im Fächerkatalog der FANG mit einem Sperrvermerk versehen. Die Inhalte der einzelnen AWPFs sind auf der fakultätseigenen Homepage der FANG veröffentlicht.		
Literatur	je nach gewählten AWPFs		

IT-Sicherheit (1512200)

IT Security

Art des Moduls Pflichtmodul	**Sprache** Deutsch	**SWS** 4	**ECTS** 5
Häufigkeit Jedes Sommersemester	**Dauer** 1 Semester	**Studiensemester** 4	**Lehr- und Lernformen** Seminaristischer Unterricht, Übung
Modulverantwortung	Prof. Dr.-Ing. Sebastian Biedermann		
Dozierende	Prof. Dr.-Ing. Sebastian Biedermann		
Verwendbarkeit			
Aufwand	*Gesamt* 150	*Präsenzzeit* 15	*Selbststudium* 135
Voraussetzungen	*nach SPO*: keine *empfohlen*: keine		
Prüfung	*Art der Prüfung*: Schriftliche Prüfung *Art der Note*: Differenzierte Note		
Lernergebnisse	Die Studierenden verstehen die grundlegenden Begriffe, Bedrohungsszenarien und Angriffsvektoren der IT-Sicherheit und können diese in unterschiedlichen Kontexten anwenden. Die Studierenden können die Sicherheitsziele Vertraulichkeit, Integrität und Verfügbarkeit erklären und deren Relevanz für IT-Systeme und Softwareprojekte bewerten. Die Studierenden kennen die Funktionsweise symmetrischer und asymmetrischer Verschlüsselungsverfahren sowie digitaler Signaturen und Zertifikate und können diese in geeigneten Szenarien einsetzen. Die Studierenden sind in der Lage, Schwachstellen in Webanwendungen (z. B. SQL-Injections) und klassischen Anwendungen (z. B. Buffer-Overflows) zu erkennen, zu analysieren und durch geeignete Maßnahmen zu beheben. Die Studierenden verstehen moderne Ansätze zur Authentifizierung und Autorisierung, wie Multi-Faktor-Authentifizierung, Rollen- und Rechtekonzepte sowie Single Sign-On (SSO), und können diese anwenden.		
Modulinhalte	Das Modul "IT-Sicherheit" vermittelt grundlegende Konzepte und praktische Ansätze zur Sicherung von IT-Systemen und Softwareanwendungen. Studierende lernen zentrale Begriffe, Bedrohungsszenarien und Angriffsvektoren kennen und verstehen die Sicherheitsziele Vertraulichkeit, Integrität und Verfügbarkeit. Ein Schwerpunkt liegt auf der Kryptografie, einschließlich symmetrischer und asymmetrischer Verschlüsselungsverfahren, digitaler Signaturen, Zertifikaten sowie Public Key Infrastrukturen (PKI) und Alternativen wie Transport-Layer-Security (TLS). Darüber hinaus werden Schwachstellen in Webanwendungen, wie SQL-Injections, sowie in klassischen Anwendungen, wie Buffer-Overflows, analysiert und Gegenmaßnahmen entwickelt. Das Modul behandelt auch Authentifizierungs- und Autorisierungskonzepte, einschließlich Multi-Faktor-Authentifizierung, Rollen- und Rechtekonzepte sowie Single Sign-On (SSO). Ein weiterer wichtiger Bestandteil ist die Bedrohungsmodellierung und Risikoanalyse, um Schwachstellen systematisch zu identifizieren und Sicherheitsmaßnahmen zu priorisieren. Praktische Übungen und Fallstudien vertiefen die theoretischen Inhalte und ermöglichen die Anwendung auf reale Szenarien. Das Modul zielt darauf ab, Studierende mit den Fähigkeiten auszustatten, IT-Systeme sicher zu gestalten und Sicherheitsstrategien effektiv in Softwareentwicklungsprozesse zu integrieren.		
Literatur	"Angewandte Kryptographie" von Wolfgang Ertel "The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws" von Dafydd Stuttard und Marcus Pinto		

	"Buffer Overflows und Format-String-Schwachstellen: Funktionsweisen, Exploits und Gegenmassnahmen" von Tobias Klein

Machine Learning (1512300)

Machine Learning

Art des Moduls	**Sprache**	**SWS**	**ECTS**
Pflichtmodul	Englisch	4	5
Häufigkeit	**Dauer**	**Studiensemester**	**Lehr- und Lernformen**
Jedes Sommersemester	1 Semester	4	Seminaristischer Unterricht, Übung
Modulverantwortung	Prof. Dr. Frank-Michael Schleif		
Dozierende	Prof. Dr. Frank-Michael Schleif		
Verwendbarkeit			
Aufwand	*Gesamt*	*Präsenzzeit*	*Selbststudium*
	150	15	135
Voraussetzungen	*nach SPO*: Data Science, Math modules, programming courses *empfohlen*: Math skills are crucial and are only briefly repeated on demand and in an initial refresher. The students should have a reasonable knowledge in programming with python		
Prüfung	*Art der Prüfung*: Portfolio *Art der Note*: Differenzierte Note		
Lernergebnisse	Upon successful completion of this module, students will be able to: • *Knowledge and Understanding** • Explain the historical development of AI and machine learning, including key symbolic and sub-symbolic approaches. • Differentiate between classical AI techniques and modern ML-based approaches. • Describe the main paradigms of machine learning and their respective application areas. • Understand the formalism of the learning problem, including key concepts such as loss functions, risk minimization, and model complexity. • Identify and explain the strengths, weaknesses, and computational characteristics of various machine learning algorithms • *Skills and Competences** • Implement and apply **core machine learning algorithms** for regression, classification, and clustering. • Utilize Python and ML libraries (Scikit-learn, Pandas, NumPy) to setup machine learning models • Evaluate and compare models using performance metrics and validation techniques • Interpret and critically assess experimental results, considering accuracy, efficiency, and model robustness. • Communicate ML findings effectively using visualizations, reports, and notebooks • Assess ethical implications and biases in machine learning models and AI applications.		
Modulinhalte	Machine learning (ML) has become a fundamental technology in modern software engineering, powering applications in **data analysis, automation, and artificial intelligence (AI)**. This module introduces students to the **core principles, models, and algorithms** of machine learning, building upon the **data science and data management** knowledge acquired in the previous semester. Students will gain an **understanding of traditional AI methods**, the **formalism of the learning problem**, and the **key machine learning paradigms**—including **supervised and unsupervised learning**. The course covers **foundational learning theories, evaluation methods, and selected ML algorithms**,		

	while also addressing **ethical and societal implications** of machine learning in modern applications. A strong **practical component** ensures that students **apply ML techniques using Python** and relevant libraries (NumPy, Pandas, Scikit-learn, Jupyter Notebooks). By the end of the course, students will be able to develop, evaluate, and optimize machine learning models, preparing them for further studies in **advanced AI, deep learning, and applied machine learning**. The module covers the following topics: • *1. Introduction to Artificial Intelligence and Machine Learning** • Refresher on linear algebra and stochastic • Historical development of AI and ML, Symbolic vs. sub-symbolic AI, Expert systems vs. statistical learning • Overview of **classical AI methods** (Adatron, Boltzmann machines, Hopfield networks, cellular automata) • *2. Core Concepts in Machine Learning** • Key **paradigms**: Supervised and unsupervised learning • Learning goals: **Prediction (regression/classification) vs. knowledge discovery (clustering/ density estimation)** • **Ethical and societal considerations** in ML • *3. Foundations of Learning from Data** • Objective (loss) functions and **empirical risk minimization** • **Overfitting vs. underfitting** and the bias-variance trade-off • Model training, validation, and testing • **Performance evaluation**: Metrics such as accuracy, precision-recall, ROC curves • *4. Key Machine Learning Algorithms** • **Linear models for regression and classification** (linear regression, logistic regression) • **Regularization techniques**: Ridge regression, LASSO • **Mixture models**: k-means clustering, Gaussian Mixture Models (GMMs) • **Non-parametric methods**: Kernel methods, decision trees, random forests • *5. Practical Implementation of ML Models** • **Programming for ML** using Python (NumPy, Pandas, Scikit-learn) • Working with **Jupyter Notebooks** for model development and documentation • Hands-on exercises in **model selection, hyperparameter tuning, and evaluation** • *6. Ethical, Legal, and Social Implications of ML** • Bias and fairness in machine learning models • Privacy considerations in AI applications • Societal impact of AI-driven decision-making
Literatur	1. Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer. 2. Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT Press. 3. Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The Elements of Statistical Learning*. Springer. 4. Russell, S., & Norvig, P. (2022). *Artificial Intelligence: A Modern Approach*. Pearson. 5. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. 6. European Commission (2023). *Ethical AI and Machine Learning Guidelines*. Further literature is provided in the course.

Mobile Systems (1512000)

Mobile Systems

Art des Moduls Pflichtmodul	**Sprache** Englisch	**SWS** 4	**ECTS** 5
Häufigkeit Jedes Sommersemester	**Dauer** 1 Semester	**Studiensemester** 4	**Lehr- und Lernformen** Seminaristischer Unterricht, Übung
Modulverantwortung	Prof. Dr. Peter Braun		
Dozierende	Prof. Dr. Peter Braun		
Verwendbarkeit			
Aufwand	*Gesamt* 150	*Präsenzzeit* 15	*Selbststudium* 135
Voraussetzungen	*nach SPO*: keine *empfohlen*: keine		
Prüfung	*Art der Prüfung*: Portfolio *Art der Note*: Differenzierte Note		
Lernergebnisse	• Understand and explain the fundamental concepts of cross-platform mobile development using Flutter and React Native, including their architecture, advantages, and limitations. • Design and implement mobile user interfaces using Flutter's widget-based system and React Native's component-based approach, ensuring usability and responsiveness. • Apply state management techniques such as Provider, Riverpod, and React Context API to efficiently manage and maintain application state in mobile applications. • Integrate mobile applications with backend services by consuming REST APIs and GraphQL, handling asynchronous data operations, and managing data caching and error handling. • Write and execute unit, widget, and integration tests to ensure application reliability using Flutter's testing framework and testing tools for React Native such as Jest and React Testing Library.		
Modulinhalte	This module introduces students to the principles and practical aspects of mobile application development. It focuses on modern cross-platform technologies and best practices for building efficient, scalable, and user-friendly mobile applications. • Introduction to Mobile User Interface Development with Flutter • Overview of Flutter and the Dart programming language • Designing responsive layouts, handling user input, and implementing navigation • State Management in Mobile Applications • Understanding stateful and stateless widgets in Flutter • Exploring state management techniques such as Provider and Riverpod • Introduction to React Native: Understanding the fundamentals of React Native and its component-based architecture • Building cross-platform mobile apps using JavaScript and React principles • Utilizing React Native components and navigation techniques • Integrating APIs and Asynchronous Data Handling in Mobile Apps • Connecting mobile applications with backend services using REST APIs and GraphQL • Handling asynchronous operations with Dart's Future & Streams and JavaScript's Promises & Async/Await • Managing data fetching, caching, and error handling • Testing and Debugging Mobile Applications		
Literatur	• Thomas Bailey, Alessandro Biessek: Flutter for Beginners: Cross-platform mobile development from Hello, World! to app release with Flutter 3.10+ and Dart 3.x. Packt, 2023.		

	• Sakhniuk, Mikhail, und Adam Boduch. React and React Native: Build Cross-platform JavaScript and TypeScript Apps for the Web, Desktop, and Mobile. Fifth edition. Birmingham, UK: Packt, 2024.

Projekt 3 (1511900)

Project 3

Art des Moduls Pflichtmodul	**Sprache** Deutsch	**SWS** 1	**ECTS** 5
Häufigkeit Jedes Sommersemester	**Dauer** 1 Semester	**Studiensemester** 4	**Lehr- und Lernformen** Projekt
Modulverantwortung	Prof. Dr. Peter Braun		
Dozierende	Prof. Dr. Peter Braun, Prof. Dr. Isabel John, Michael Rott, Prof. Dr. Rolf Schillinger, Prof. Dr.-Ing. Tobias Fertig, Prof. Dr. Frank-Michael Schleif, Prof. Dr.-Ing. Sebastian Biedermann, Prof. Dr. Tristan Wimmer, Prof. Dr.-Ing. Anne Heß		
Verwendbarkeit			
Aufwand	*Gesamt* 150	*Präsenzzeit* 15	*Selbststudium* 135
Voraussetzungen	*nach SPO*: keine *empfohlen*: keine		
Prüfung	*Art der Prüfung*: Dokumentation, Präsentation *Art der Note*: Differenzierte Note		
Lernergebnisse	• Ein verteiltes Software-System mit modernen Technologien zu entwickeln • IT-Sicherheitsaspekte in die Software-Architektur zu integrieren • Komplexe Softwarelösungen in Teams zu konzipieren und umzusetzen • Verschiedene Rollen in einem agilen Team zu übernehmen • Fortgeschrittene Methoden des agilen Projektmanagement anzuwenden, insbesondere Reviews und Retrospektiven durchzuführen • Eine vollständige Software-Dokumentation zu erstellen • Ihr Projekt in einer öffentlichen Abschlusspräsentation überzeugend darzustellen		
Modulinhalte	Die Studierenden arbeiten in größeren Teams (5–6 Personen) an einem komplexen Softwareprojekt mit echten funktionalen und nicht-funktionalen Anforderungen, die von einem Kunden durch Interviews zu ermitteln sind. Die Studierenden entwickeln sie ein verteiltes Softwaresystem, das moderne Software Engineering Techniken integriert. Das Team organisiert seine Aufgaben durch eine agile Projektmanagement-Methodik und verteilt die Rollen im Team entsprechend. Die Anwendung könnte enthalten: • Microservices oder eine verteilte Architektur • Datenbankanbindung (SQL oder NoSQL) • Eine grafische Benutzeroberfläche, zum Beispiel als Web- oder Mobile Anwendung • Sicherheitsaspekte (z. B. Authentifizierung, Berechtigungen, Verschlüsselung) • Performanz- und Skalierbarkeitsoptimierung • Automatisierte Unit- und Integrations-Tests zur Qualitätssicherung • Die Dokumentation folgt Industrie-Standards mit vollständiger Architektur-, API- und Deployment-Dokumentation.		
Literatur			

Web Systems (1512100)

Web Systems

Art des Moduls Pflichtmodul	**Sprache** Deutsch	**SWS** 4	**ECTS** 5
Häufigkeit Jedes Sommersemester	**Dauer** 1 Semester	**Studiensemester** 4	**Lehr- und Lernformen**
Modulverantwortung	Prof. Dr. Rolf Schillinger		
Dozierende	Prof. Dr. Rolf Schillinger		
Verwendbarkeit			
Aufwand	*Gesamt* 150	*Präsenzzeit* 15	*Selbststudium* 135
Voraussetzungen	*nach SPO*: keine *empfohlen*: keine		
Prüfung	*Art der Prüfung*: Schriftliche Prüfung *Art der Note*: Differenzierte Note		
Lernergebnisse	Nach erfolgreichem Abschluß dieses Moduls sind die Studierenden in der Lage • HTML Templating Systeme und deren wichtigste Eigenschaften voneinander abzugrenzen und sinnvolle Auswahlen zu treffen • Vorgegebene HTML Layouts in Template Syteme zu übertragen • eine reaktive Webapplikation zu designen und als SPA in React umzusetzen • die entstandene Webapplikation auf deren Rendering Performanz hin zu untersuchen und zu optimieren • die JWT und OAuth2 Spezifikationen im Detail zu verstehen und auf dieser Basis zu entscheiden, welche Lösung für vorgegebene Anforderung verwendet werden kann • beide Varianten in einer React Beispielanwendung aktiv zu verwenden		
Modulinhalte	Web Systeme haben in den letzten Jahrzehnten viele klassische GUI Anwendungen ersetzt, die vormals unter Nutzung von GUI Frameworks wie MFC/.NET/WPF, Qt, Swing und ähnlichem programmiert wurden. Vielen Vorteilen wie zum Beispiel der sehr großer Platformunabhängigkeit oder der problemlosen Ausrollung auf eine beliebige Anzahl von Clients, standen immer wieder auch erhebliche Nachteile gegenüber. So waren Web Systeme lange Zeit nur online nutzbar oder transportieren kein systemtypisches CI, wie es z.B. WPF für Windows bietet. In der Praxis überwiegen in der Mehrzahl der Anwendungsfälle allerdings mittlerweile die Vorteile. In diese Modul lernen Studierende die verschiedenen Ansätze und Technologien kennen und nutzen, auf denen moderne Web Systeme basieren. Dazu gehören insbesondere die folgenden Themen: • Grundlagen • Klassische HTML Templating Systeme wie Blade in PHP oder Django in Python • Reaktivität in Webapplikationen, dabei insbesondere reaktive JavaScript Frameworks wie React • HTML DOM und Rendering Prozesse im Browser • Single Page Web Apps • Progressive Web Apps • Optimierung des Page Renderings • Critical Rendering Path • Serverside Rendering • Authentifizierung und Autorisierung • JWT und OAuth2 • FIDO2 (Passkeys) • Ausblick • WebAssembly		

	• Electron Apps
Literatur	• Mozilla Developer Network (https://developer.mozilla.org/de/) • Banks, A., & Porcello, E. (2020). Learning React: modern patterns for developing React apps. O'Reilly Media. • Google SREs Building Secure and Reliable Systems (https://google.github.io/building-secure-and-reliable-systems/raw/toc.html)

Semester 5

Praxismodul (1512500)

Internship Semester

Art des Moduls Pflichtmodul	**Sprache** Deutsch/Englisch	**SWS** 1	**ECTS** 30
Häufigkeit Jedes Semester	**Dauer** 1 Semester	**Studiensemester** 5	**Lehr- und Lernformen**
Modulverantwortung	Prof. Dr. Peter Braun		
Dozierende			
Verwendbarkeit			
Aufwand	*Gesamt* 150	*Präsenzzeit* 15	*Selbststudium* 135
Voraussetzungen	*nach SPO*: Mehr als 90 ECTS Punkte, davon 55 ECTS aus Modulen aus dem ersten Studienjahr sowie das Modul Professional Skills ist bestanden. *empfohlen*: Zusätzlich zu den genannten Voraussetzungen insbesondere auch: Software Qualität, Backend Systems, Mobile Systems, Web Systems,		
Prüfung	*Art der Prüfung*: Präsentation, Dokumentation *Art der Note*: ME/OE		
Lernergebnisse	Die Studierenden sollen während des Praxissemester folgende Lernziele erreichen: • Praxisorientierte Kenntnisse betrieblicher Abläufe und Prozesse zu erwerben und deren Relevanz für die Softwareentwicklung zu verstehen. • Selbstständig und eigenverantwortlich in IT-Projekten zu arbeiten, unter Anleitung und mit zunehmender Eigeninitiative. • Die im Studium erworbenen theoretischen und praktischen Kompetenzen in realen Projekten anzuwenden und mit den Erfahrungen der Praxis zu verknüpfen. • Anforderungen aus der Praxis (z. B. Kundenwünsche, technische Anforderungen) zu analysieren und deren Bedeutung für die Entwicklung und Umsetzung von Softwarelösungen zu verstehen. • Problemlösungen für betriebliche Prozesse oder IT-Projekte zu entwerfen, zu implementieren und deren Wirksamkeit zu bewerten. • Effektiv im Team zu arbeiten, Rollenverteilungen zu verstehen und kollaborative Entwicklungsprozesse aktiv mitzugestalten. • Die Einbettung von Softwareprojekten in die organisatorischen Strukturen eines Unternehmens zu erkennen und aktiv in betriebliche Abläufe zu integrieren. • Das Berufsfeld eines Software Engineers in einem professionellen Umfeld kennenzulernen und Erfahrungen für die eigene Karriereplanung zu sammeln. • Bei Problemen eigenständig die richtigen Ansprechpartner im Unternehmen zu identifizieren und mit ihnen zielführend zu kommunizieren. • Best Practices für professionelle Softwareentwicklung zu erleben, den hohen Qualitätsanspruch in Unternehmen nachzuvollziehen und Exzellenz als Zielsetzung zu verinnerlichen. • Die Dynamik und Motivation in professionellen Entwicklungsteams zu erleben und die Bedeutung von Kommunikation, Verantwortung und Teamkultur für den Projekterfolg zu erkennen. • Den Sinn und die Wirkung der eigenen Tätigkeit im Unternehmen zu reflektieren und den Beitrag zur Wertschöpfung und zum Gesamterfolg der Organisation zu erkennen.		
Modulinhalte	Im Rahmen des Praxismoduls erhalten die Studierenden die Möglichkeit, ihre im Studium erworbenen fachlichen und methodischen Kompetenzen in einem realen Unternehmensumfeld anzuwenden und zu vertiefen. Die Praxisphase umfasst ein größeres IT-Projekt, in dem die Studierenden möglichst viele Phasen des Softwareentwicklungsprozesses eigenverantwortlich durchlaufen, darunter: • Anforderungsanalyse und Systemspezifikation		

	• Softwarearchitektur und -design • Implementierung und Testen • Systemintegration und -einführung • Dokumentation und Qualitätssicherung • Das Projekt soll einen zeitlichen Umfang von mindestens 12 Wochen haben und praxisnahe Problemstellungen aus dem Bereich Software Engineering adressieren. Zusätzlich wird empfohlen, dass die Studierenden vor dem Projekt verschiedene Abteilungen und Unternehmensbereiche kennenlernen, um ein ganzheitliches Verständnis für betriebliche Prozesse und die Zusammenarbeit in interdisziplinären Teams zu entwickeln. Die Praxisphase wird durch die Hochschule begleitet. Ansprechpartner für die begleitete Praxisphase ist Prof. Dr. Peter Braun.
Literatur	

Semester 6

Advanced Software Testing (1513000)

Advanced Software Testing

Art des Moduls Pflichtmodul	**Sprache** Englisch	**SWS** 4	**ECTS** 5
Häufigkeit Jedes Sommersemester	**Dauer** 1 Semester	**Studiensemester** 6	**Lehr- und Lernformen** Seminaristischer Unterricht, Übung
Modulverantwortung	Prof. Dr. Tristan Wimmer		
Dozierende	Prof. Dr. Steffen Heinzl, Prof. Dr. Tristan Wimmer		
Verwendbarkeit			
Aufwand	*Gesamt* 150	*Präsenzzeit* 15	*Selbststudium* 135
Voraussetzungen	*nach SPO*: keine *empfohlen*: Programmieren 1 und 2, Introduction to Software Engineering, Analyse und Design, Software Qualität		
Prüfung	*Art der Prüfung*: Portfolio, Schriftliche Prüfung *Art der Note*: Differenzierte Note		
Lernergebnisse	Die Studierenden verstehen die grundlegenden Konzepte und Prinzipien des Softwaretestens, einschließlich Testarten, Testabdeckung und Testpyramide. Die Studierenden analysieren verschiedene Testarchitekturen und bewerten deren Eignung für unterschiedliche Softwareprojekte. Die Studierenden wenden Testautomatisierungsstrategien an und implementieren automatisierte Tests mit JUnit, Selenium und Wiremock. Die Studierenden entwickeln eine strukturierte Testarchitektur unter Berücksichtigung von SOLID-Prinzipien und Design Patterns. Die Studierenden erstellen Behaviour Driven Development (BDD)-Tests mit Cucumber & Gherkin und binden sie in den Entwicklungsprozess ein. Die Studierenden integrieren automatisierte Tests in einen DevOps-Prozess und setzen Continuous Testing mit Jenkins um. Die Studierenden bewerten die Qualität von Tests anhand von Metriken wie Testabdeckung, Robustheit und Wartbarkeit. Die Studierenden experimentieren mit explorativen Testmethoden und wenden geeignete Techniken zur Fehlererkennung an.		
Modulinhalte	Grundlagen des Testens Motivation und Ziele von Softwaretests Testarten: Black-, White- & Grey-Box-Testing Funktionale & nicht-funktionale Tests Testabdeckung, Testpfade & Testpyramide Testautomatisierung Einführung in automatisierte Tests und Erfolgsfaktoren Testframeworks: JUnit, Maven, Annotationen & Assertions Testarten: Record Replay, Scripted & Keyword-driven Testing Testarchitektur & Design Patterns SOLID-Prinzipien für Testarchitekturen 4-Schichten-Testkonzept: Modellierung, Definition, Execution, Adaptation Wichtige Design Patterns für Testing Automatisiertes UI- & API-Testing Einführung in Selenium: Driver, PageObject Pattern, Identifikatoren Mocking mit Wiremock für API-Tests Behaviour Driven Development (BDD) Einführung in Cucumber & Gherkin		

	Feature-Files, Step-Files & Szenario Outlines Exploratives Testen & DevOps-Integration Explorative Testmethoden und Techniken Continuous Testing mit Jenkins & DevOps Pipelines
Literatur	Essentials of Software Testing von Ralf Bierig, Stephen Brown, Edgar Galván, Joe Timoney, 2021, Cambridge University Press

Clean Code und Design Pattern (1512900)

Clean Code and Design Pattern

Art des Moduls	**Sprache**	**SWS**	**ECTS**
Pflichtmodul	Deutsch	4	5
Häufigkeit	**Dauer**	**Studiensemester**	**Lehr- und Lernformen**
Jedes Sommersemester	1 Semester	6	Seminaristischer Unterricht, Übung
Modulverantwortung	Prof. Dr.-Ing. Tobias Fertig		
Dozierende	Prof. Dr. Steffen Heinzl, Prof. Dr.-Ing. Tobias Fertig		
Verwendbarkeit			
Aufwand	*Gesamt*	*Präsenzzeit*	*Selbststudium*
	150	15	135
Voraussetzungen	*nach SPO*: Die Studierenden sollen ein eigenes Projekt aus bspw. dem Programmierprojekt, Softwareentwicklungsprojekt, Praktikum, o.ä. mitbringen. *empfohlen*: keine		
Prüfung	*Art der Prüfung*: Portfolio, Schriftliche Prüfung *Art der Note*: Differenzierte Note		
Lernergebnisse	Nach dem erfolgreichen Abschluss des Moduls sind die Studierenden in der Lage • verständlicheren Code zu schreiben • besser pflegbaren Code zu schreiben • weniger fehleranfälligen Code zu schreiben • an geeigneten Code-Stellen zu formulieren, WAS erreicht werden soll und nicht WIE • Crosscutting Concerns zu verstehen • Refactoring durchzuführen • Design Patterns zu verstehen und anzuwenden		
Modulinhalte	Dieses Modul vermittelt weit-verbreitete Design Patterns und Clean Code Techniken. Clean Code Regeln zu kennen ist essentiell für die Code-Qualität und Wartbarkeit. Deshalb werden in diesem Modul folgende Inhalte behandelt: • Naming, Functions, Comments, Formatting • Objects and Data Structures • Law of Demeter • DTOs • Exceptions, Don't return null • Code Tangling and Scattering, Logging, AOP • Fluent Interfaces, Internal DSLs • Coding Dojos • Dependency Injection • Traditionelle Design Patterns • Moderne Web und Mobile Design Patterns • Refactoring Strategien		
Literatur	Robert C. Martin (2008). Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall. Gamma, E., Helm, R., Johnson, R., Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Pearson Education. Fowler, M., Beck, K., Brant, J., Opdyke, W., Roberts, D. (2012). Refactoring: Improving the Design of Existing Code. Pearson Education.		

Cloud Computing (1512800)

Cloud Computing

Art des Moduls	**Sprache**	**SWS**	**ECTS**
Pflichtmodul	Deutsch	4	5
Häufigkeit	**Dauer**	**Studiensemester**	**Lehr- und Lernformen**
Jedes Sommersemester	1 Semester	6	Seminaristischer Unterricht, Übung
Modulverantwortung	Prof. Dr. Rolf Schillinger		
Dozierende	Prof. Dr. Rolf Schillinger		
Verwendbarkeit			
Aufwand	*Gesamt*	*Präsenzzeit*	*Selbststudium*
	150	15	135
Voraussetzungen	*nach SPO*: keine *empfohlen*: keine		
Prüfung	*Art der Prüfung*: Portfolio *Art der Note*: Differenzierte Note		
Lernergebnisse	Nach erfolgreichem Abschluß des Moduls sind die Studiernden in der Lage • die verschiedenen Cloud Betreibermodelle verstehen und voneinander abgrenzen zu können • für gegebene Applikationen geeignete Cloud Technologien auszuwählen und deren Nutzung zu planen • eine VM auf einer bereitgestellten Proxmox Instanz zu erstellen, zu konfigurieren und nach Installation der benötigten Software (Apache + DB) darauf eine Backend Applikation bereitzustellen • die Backend Applikation von der VM zu lösen und per LXC anzubieten • die Backend Applikation in Docker Container zu verpacken und manuell zu deployen • diese Container mittels Kubernetes skalierbar zu orchestrieren • Methoden des load testings anzuwenden um das Verhalten des Deployments unter verschiedenen Lastprofilen abschätzen zu können		
Modulinhalte	Seit den ersten Anfängen der Virtualiiserung in IBMs Mainframe Betriebssystemen der 80er Jahre haben sich Technologien der Virtualiserung zum heutigen Cloud Computing Komplex hin entwickelt, der für einige bahnbrechende Neuerungen im Betrieb von IT-Services gesorgt hat. Viele der heute von modernen Rechenzentren geforderten Eigenschaften wie größte Flexibilität, Agilität und Skalierbarkeit waren vor der großflächigen Verfügbarkeit von Cloud Technologien nicht oder nur sehr schwer umsetzbar. Dieser Kurs beschäftigt sich mit den wichtigsten Grundlagen, Funktionalitäten und Eigenschaften des Cloud Computings, insbesondere mit den folgenden Teilgebieten: • Grundlagen und Abgrenzung - Virtualisierung als Grundlage des Cloud Computings - Cloud Betreibermodelle und deren Charakteristika - GigaScaler (AWS, Azure, Goolge Cloud Platform) • Deployment in Cloud Umgebungen - Cloud im Context DevOps - Automatisierung des Deployments (Pipelines, CI/CD) - Skalierungsmöglichkeiten • Virtuelle Maschinen - Überblick über gängige Virtualisierungstechnologien (KVM / QEMU, Proxmox, Hyper-V) - Infrastructure as code - VM Introspection / Sicherheit von VMs • Containerisierung - BSD Jails, LXC		

	- Docker Container - Orchestrierung von Containern (Kubernetes) • Überblick über die PaaS Angebote einiger GigaScalerPlatform as a Service
Literatur	• Tanenbaum, A. & Bos, H. (2014). Modern Operating Systems. (4th ed.). Pearson International. https://elibrary.pearson.de/book/99.150005/9781292061955 • Stender, D. (2020). Cloud-Infrastrukturen: Das Handbuch für DevOps-Teams und Administratoren. Rheinwerk Verlag. • Liebel, O. (2023). Skalierbare Container-Infrastrukturen: Das Handbuch für Planung und Administration. Rheinwerk Verlag.

Datenschutz und Ethik (1512700)

Data Protection and Ethics

Art des Moduls Pflichtmodul	**Sprache** Deutsch	**SWS** 4	**ECTS** 5
Häufigkeit Jedes Sommersemester	**Dauer** 1 Semester	**Studiensemester** 6	**Lehr- und Lernformen** Seminaristischer Unterricht, Übung
Modulverantwortung	Prof. Dr. Markus Oermann		
Dozierende	Prof. Dr. Markus Oermann		
Verwendbarkeit			
Aufwand	*Gesamt* 150	*Präsenzzeit* 15	*Selbststudium* 135
Voraussetzungen	*nach SPO*: keine *empfohlen*: keine		
Prüfung	*Art der Prüfung*: Schriftliche Prüfung *Art der Note*: Differenzierte Note		
Lernergebnisse	Im Zuge ihrer Teilnahme am Modul • gewinnen die Studierenden einen Überblick über ethische Anforderungen an die Softwareentwicklung und lernen, wie sich diese in Arbeitsprozessen abbilden lassen • erwerben die Studierenden Kenntnisse der Grundstrukturen des Datenschutzrechts und können Grundfragen zur Datenschutzcompliance beantworten • erwerben die Studierenden Kenntnisse der Grundstrukturen des Rechtsrahmens für Künstliche Intelligenz • werden die Studierenden kommunikations- und dialogfähig mit den entsprechenden Expertinnen und Experten für rechtliche Fragestellungen in ihrem späteren Arbeitsumfeld		
Modulinhalte	Im Abschnitt zu Ethik werden essenzielle begriffliche Grundlagen der Moralphilosophie erläutert. Auf der Grundlage etablierter Schulen der Ethik wird die normative Begründung und anhand von problematischen Praxisfällen der tatsächliche Bedarf für eine ethische fundierte Softwareentwicklung und einen entsprechenden Ethos der Verantwortlichen herausgearbeitet. Die Betrachtung von Modellen für die Integration ethischer Überlegungen in Entwicklungs- und Systemdesignprozesse verdeutlicht, wie ethischen Grundsätze in der Praxis in die Softwareentwicklung einfließen können. In der Praxis sind zudem Fragen der Compliance mit dem geltenden Datenschutzrecht von besonderer Relevanz. Nach einem Überblick über dessen Grundstrukturen liegt der Schwerpunkt auf den Anforderungen an den technischen und organisatorischen Datenschutz sowie der Durchsetzung und den Folgen von Rechtsverstößen. Hier fließen auch relevante Aspekte des Informationssicherheitsrechts ein. Abschließend wird ein Überblick über den europäischen Rechtsrahmen für künstliche Intelligenz vermittelt.		
Literatur	Schweppenhäuser, Gerhard (2021): Grundbegriffe der Ethik. Reclam, Ditzingen: Kap. 2.3 - 3. v. Lewinski/Rüpke/Eckhardt (2022): Datenschutzrecht. 2. Auflage. München, C.H. Beck. Windholz, Natascha et al. (2024): Praxishandbuch KI-VO. 1 Auflage. München, Hanser. (bis zur erstmaligen Durchführung nochmals zu aktualisieren)		

Projektarbeit (1512600)

Software Development Project

Art des Moduls	**Sprache**	**SWS**	**ECTS**
Pflichtmodul	Deutsch/Englisch	1	10
Häufigkeit	**Dauer**	**Studiensemester**	**Lehr- und Lernformen**
Jedes Semester	1 Semester	6	Projekt
Modulverantwortung	Prof. Dr. Peter Braun		
Dozierende	Prof. Dr. Peter Braun, Prof. Dr. Isabel John, Michael Rott, Prof. Dr. Rolf Schillinger, Prof. Dr.-Ing. Tobias Fertig, Prof. Dr. Frank-Michael Schleif, Prof. Dr.-Ing. Sebastian Biedermann, Prof. Dr. Tristan Wimmer, Prof. Dr.-Ing. Anne Heß		
Verwendbarkeit			
Aufwand	*Gesamt* 300	*Präsenzzeit* 15	*Selbststudium* 285
Voraussetzungen	*nach SPO*: keine *empfohlen*: keine		
Prüfung	*Art der Prüfung*: Projektarbeit *Art der Note*: Differenzierte Note		
Lernergebnisse	Nach Abschluss des Moduls sind die Studierenden in der Lage: • Ein komplexes Software-System mit modernen Technologien zu entwickeln • Eine wissenschaftliche Fragestellung aus dem Bereich Software Engineering zu bearbeiten und methodisch zu evaluieren • IT-Sicherheitsaspekte in die Software-Architektur zu integrieren • Mit externen Stakeholdern aus Industrie oder Forschung zu arbeiten und deren Anforderungen zu erheben • Methoden des agilen Projektmanagement anzuwenden, inkl. Zeitschätzung und Zeiterfassung • Eine vollständige wissenschaftliche Software-Dokumentation nach akademischen Standards zu erstellen • Ergebnisse auf einer Konferenz oder in einem wissenschaftlichen Kolloquium zu präsentieren		
Modulinhalte	Die Studierenden arbeiten in größeren Teams (5–6 Personen) an einem wissenschaftlich-motivierten und/oder praxisnahen Softwareprojekt mit komplexen funktionalen und nicht-funktionalen Anforderungen. Diese Anforderungen werden entweder durch Interviews mit einem externen Stakeholder (z. B. Unternehmen oder Forschungspartner) oder durch wissenschaftliche Fragestellungen aus aktuellen Forschungsthemen ermittelt. Das Projekt beinhaltet sowohl die technische Umsetzung als auch eine wissenschaftliche Evaluierung der getroffenen Entscheidungen. Die Anwendung könnte folgende Elemente enthalten: • Moderne Softwarearchitektur, z. B. Microservices oder verteilte Systeme • Datenbankanbindung (SQL oder NoSQL) mit optimierter Abfrageperformance • Eine grafische Benutzeroberfläche (Web oder Mobile) mit Usability-Tests • Sicherheitsaspekte (z. B. Authentifizierung, Verschlüsselung, Security Audits) • Wissenschaftliche Evaluierung von Architekturentscheidungen, Algorithmen oder Technologien • Zusammenarbeit mit externen Stakeholdern (Industriepartner oder Forschungseinrichtung) • Automatisierte Tests und Code-Qualitätsanalysen zur Sicherstellung langfristiger Wartbarkeit • Erstellung eines wissenschaftlichen Abschlussberichts oder Konferenzbeitrags • Das Projekt schließt mit einer öffentlichen Abschlusspräsentation, in der die wissenschaftlichen Erkenntnisse sowie die entwickelte Software demonstriert werden.		

Literatur	

Semester 7

Bachelorarbeitsmodul (1513400)

Bachelor Thesis / Bachelor Seminar

Art des Moduls Pflichtmodul	**Sprache** Deutsch/Englisch	**SWS** 1	**ECTS** 15
Häufigkeit Jedes Semester	**Dauer** 1 Semester	**Studiensemester** 7	**Lehr- und Lernformen** Seminar
Modulverantwortung	Prof. Dr. Peter Braun		
Dozierende	Prof. Dr. Peter Braun, Prof. Dr. Isabel John, Prof. Dr. Eva Wedlich, Prof. Dr. Rolf Schillinger, Prof. Dr.-Ing. Tobias Fertig, Prof. Dr. Frank-Michael Schleif, Prof. Dr.-Ing. Sebastian Biedermann, Prof. Dr. Tristan Wimmer, Prof. Dr.-Ing. Anne Heß		
Verwendbarkeit	Bachelor Wirtschaftsinformatik		
Aufwand	*Gesamt* 450	*Präsenzzeit* 40	*Selbststudium* 410
Voraussetzungen	*nach SPO*: 120 ECTS-Punkte aus den ersten vier Semestern, Praxismodul, Projektarbeit *empfohlen*: keine		
Prüfung	*Art der Prüfung*: Thesis, Präsentation *Art der Note*: Differenzierte Note		
Lernergebnisse	Mit der Bachelorarbeit / dem Bachelorseminar erbringen die Studierenden den Nachweis, dass sie fähig sind, selbständig eine anspruchsvolle Aufgabenstellung aus dem Gebiet Software Engineering (ggf. fächerübergreifend) zu lösen und dass sie dabei die methodischen und wissenschaftlichen Grundlagen des Faches beherrschen sowie das Ergebnis adäquat darstellen können.		
Modulinhalte	Das Bachelorarbeitsmodul setzt sich zusammen aus der Bachelorarbeit (12 CP) sowie dem Bachelorseminar (3 CP). Die Bachelorarbeit umfasst eigene Studien und Recherchen über den Stand der Technik und der Wissenschaft des jeweiligen Themengebiets. Die Arbeit muss von Randbedingungen abstrahieren, die ihrer Natur nach nicht technisch begründet sind, sondern aus den spezifischen Gegebenheiten des Unternehmens resultieren. Soweit softwaretechnische Lösungen als Teil der Aufgabe gefordert sind, heißt das in der Regel, dass Prototypen implementiert werden, nicht aber die Sicherstellung von Produkteigenschaften (inkl. begleitender Handbücher etc.) eingeschlossen ist. Im Bachelorseminar werden die Grundzüge des wissenschaftlichen Arbeitens vermittelt und geübt.		
Literatur	in Abhängigkeit des gestellten Themas; wissenschaftliche Literatur ist entsprechend des Themas intensiv zu sichten, zu verwenden und zu zitieren		

FWPM 1 (5003xxx)

FWPM 1

Art des Moduls Wahlpflichtmodul	**Sprache** Deutsch/Englisch	**SWS** 4	**ECTS** 5
Häufigkeit Jedes Wintersemester	**Dauer** 1 Semester	**Studiensemester** 7	**Lehr- und Lernformen** Seminar
Modulverantwortung	Prof. Dr. Peter Braun		
Dozierende			
Verwendbarkeit			
Aufwand	*Gesamt* 150	*Präsenzzeit* 15	*Selbststudium* 135
Voraussetzungen	*nach SPO*: keine *empfohlen*: keine		
Prüfung	*Art der Prüfung*: Schriftliche Prüfung *Art der Note*: Differenzierte Note		
Lernergebnisse	Nach erfolgreichem Abschluss der Wahlveranstaltung sind die Studierenden in der Lage: • Spezialisierte Konzepte und Technologien in einem ausgewählten Bereich des Software Engineering zu verstehen, anzuwenden und kritisch zu reflektieren, um komplexe Problemstellungen gezielt zu lösen. • Eigenständig moderne Softwarelösungen zu entwerfen und zu implementieren, unter Berücksichtigung von Best Practices, aktuellen Entwicklungsmethoden und Qualitätsstandards. • Innovative Ansätze und aktuelle Forschungsergebnisse aus dem gewählten Themenbereich zu analysieren und deren praktische Relevanz für die Softwareentwicklung zu bewerten.		
Modulinhalte	Die Wahlveranstaltungen ermöglichen den Studierenden eine fachliche Vertiefung in spezifischen Bereichen des Software Engineering. Dabei stehen aktuelle Technologien, Methoden und Konzepte im Fokus, die praxisnah vermittelt und in Projekten angewendet werden. Die inhaltliche Ausgestaltung der Wahlveranstaltungen variiert je nach angebotenen Themen, kann jedoch folgende Schwerpunkte umfassen: • Moderne Softwarearchitekturen: Vertiefung in Microservices, Event-Driven Architecture, Domain-Driven Design (DDD) oder Cloud-native Entwicklung • Fortgeschrittene Programmierkonzepte: Funktionale Programmierung, Metaprogrammierung, Compilerbau oder domänenspezifische Sprachen (DSLs) • Softwarequalität und Teststrategien: Testautomatisierung, Continuous Integration/ Continuous Deployment (CI/CD), statische Codeanalyse und Performance-Optimierung • Software Security: Sichere Softwareentwicklung, Penetration Testing, Sicherheitskonzepte in verteilten Systemen • Mobile und Web-Technologien: Entwicklung von Progressive Web Apps (PWA), native vs. hybride App-Entwicklung, moderne Frameworks und UI/UX-Optimierung • KI und maschinelles Lernen im Software Engineering: Grundlagen des maschinellen Lernens, Einsatz von KI für Softwaretests oder Codegenerierung • Datenbank- und Persistenztechnologien: NoSQL-Datenbanken, verteilte Datenhaltung, Optimierung von Datenbankabfragen Die Studierenden setzen sich in den Wahlveranstaltungen intensiv mit aktuellen Herausforderungen und Lösungen in der Softwareentwicklung auseinander. Praktische Übungen, Projektarbeiten und Diskussionen ergänzen die theoretischen Inhalte und fördern eine anwendungsorientierte Auseinandersetzung mit den jeweiligen Themenbereichen.		
Literatur	Abhängig vom gewählte Modul.		

FWPM 2 (5003xxx)

FWPM 2

Art des Moduls Wahlpflichtmodul	**Sprache** Deutsch/Englisch	**SWS** 4	**ECTS** 5
Häufigkeit Jedes Wintersemester	**Dauer** 1 Semester	**Studiensemester** 7	**Lehr- und Lernformen** Seminar
Modulverantwortung	Prof. Dr. Peter Braun		
Dozierende			
Verwendbarkeit			
Aufwand	*Gesamt* 150	*Präsenzzeit* 15	*Selbststudium* 135
Voraussetzungen	*nach SPO*: keine *empfohlen*: keine		
Prüfung	*Art der Prüfung*: Schriftliche Prüfung *Art der Note*: Differenzierte Note		
Lernergebnisse	Nach erfolgreichem Abschluss der Wahlveranstaltung sind die Studierenden in der Lage: • Spezialisierte Konzepte und Technologien in einem ausgewählten Bereich des Software Engineering zu verstehen, anzuwenden und kritisch zu reflektieren, um komplexe Problemstellungen gezielt zu lösen. • Eigenständig moderne Softwarelösungen zu entwerfen und zu implementieren, unter Berücksichtigung von Best Practices, aktuellen Entwicklungsmethoden und Qualitätsstandards. • Innovative Ansätze und aktuelle Forschungsergebnisse aus dem gewählten Themenbereich zu analysieren und deren praktische Relevanz für die Softwareentwicklung zu bewerten.		
Modulinhalte	Die Wahlveranstaltungen ermöglichen den Studierenden eine fachliche Vertiefung in spezifischen Bereichen des Software Engineering. Dabei stehen aktuelle Technologien, Methoden und Konzepte im Fokus, die praxisnah vermittelt und in Projekten angewendet werden. Die inhaltliche Ausgestaltung der Wahlveranstaltungen variiert je nach angebotenen Themen, kann jedoch folgende Schwerpunkte umfassen: • Moderne Softwarearchitekturen: Vertiefung in Microservices, Event-Driven Architecture, Domain-Driven Design (DDD) oder Cloud-native Entwicklung • Fortgeschrittene Programmierkonzepte: Funktionale Programmierung, Metaprogrammierung, Compilerbau oder domänenspezifische Sprachen (DSLs) • Softwarequalität und Teststrategien: Testautomatisierung, Continuous Integration/ Continuous Deployment (CI/CD), statische Codeanalyse und Performance-Optimierung • Software Security: Sichere Softwareentwicklung, Penetration Testing, Sicherheitskonzepte in verteilten Systemen • Mobile und Web-Technologien: Entwicklung von Progressive Web Apps (PWA), native vs. hybride App-Entwicklung, moderne Frameworks und UI/UX-Optimierung • KI und maschinelles Lernen im Software Engineering: Grundlagen des maschinellen Lernens, Einsatz von KI für Softwaretests oder Codegenerierung • Datenbank- und Persistenztechnologien: NoSQL-Datenbanken, verteilte Datenhaltung, Optimierung von Datenbankabfragen Die Studierenden setzen sich in den Wahlveranstaltungen intensiv mit aktuellen Herausforderungen und Lösungen in der Softwareentwicklung auseinander. Praktische Übungen, Projektarbeiten und Diskussionen ergänzen die theoretischen Inhalte und fördern eine anwendungsorientierte Auseinandersetzung mit den jeweiligen Themenbereichen.		
Literatur	Abhängig vom gewählten Modul.		

Green IT (1513300)

Green IT

Art des Moduls	**Sprache**	**SWS**	**ECTS**
Pflichtmodul	Deutsch	4	5
Häufigkeit	**Dauer**	**Studiensemester**	**Lehr- und Lernformen**
Jedes Wintersemester	1 Semester	7	Seminaristischer Unterricht, Übung
Modulverantwortung	Prof. Dr. Frank-Michael Schleif		
Dozierende	Prof. Dr. Frank-Michael Schleif		
Verwendbarkeit			
Aufwand	*Gesamt*	*Präsenzzeit*	*Selbststudium*
	150	15	135
Voraussetzungen	*nach SPO*: Given the advanced stage in the curriculum, students are expected to bring prior knowledge in software engineering, cloud computing, and machine learning, allowing for a deeper examination of sustainability challenges and solutions in IT. Programming experience (e.g., Python) and familiarity with AI/ML frameworks (e.g., TensorFlow, PyTorch) are recommended but not mandatory. *empfohlen*: • Software Engineering • Machine Learning • Data Science		
Prüfung	*Art der Prüfung*: Portfolio, Schriftliche Prüfung *Art der Note*: Differenzierte Note		
Lernergebnisse	Knowledge and Understanding: • Explain the fundamental principles of **Green IT** and **Green AI**, including their environmental and economic implications. • Describe energy-efficient **IT infrastructures**, including sustainable data centers, hardware choices, and cloud computing strategies. • Understand the impact of **machine learning and AI models** on resource consumption and strategies for optimizing their efficiency. • Identify and interpret relevant **legal regulations, policies, and standards** related to sustainable IT and AI (e.g., EU AI Act, ISO 14001). • Discuss process modeling approaches for **sustainability in IT systems** and their role in optimizing energy and resource use. Skills and Competences: • Assess the **carbon footprint of IT infrastructures and AI models** using established evaluation tools. • Apply **sustainable software engineering** principles, including energy-aware programming and efficient algorithm design. • Optimize **AI models and workflows** through techniques such as quantization, pruning, and federated learning. • Analyze and model IT business processes from a **sustainability perspective**, identifying opportunities for optimization. • Develop recommendations for sustainable IT strategies in companies and organizations, considering **technical, regulatory, and economic aspects**.		
Modulinhalte	The increasing digitization of all sectors comes with significant environmental impacts, particularly through energy consumption and resource-intensive computing infrastructures. This module explores concepts and methods for making IT systems more sustainable, covering both classical **Green IT** aspects—such as energy-efficient hardware, data centers,		

	and software engineering—and the emerging field of **Green AI**. The latter focuses on optimizing machine learning models, improving efficiency in AI-driven processes, and minimizing carbon footprints. A further emphasis is placed on **process modeling and optimization** in the context of sustainable IT management, as well as on relevant **legal and regulatory frameworks** guiding environmentally conscious digital transformation. The module integrates theoretical foundations with practical strategies, equipping students with knowledge and skills to develop, implement, and evaluate sustainable IT and AI solutions. Some parts of the topics will also be aligned with enrichment lectures given by relevant experts. <Course Content> The module is structured around the following core topics: 1. **Foundations of Green IT** • The environmental impact of digital technologies • Principles of energy-efficient computing • Sustainable IT infrastructures: data centers, cloud computing, and hardware choices 2. **Green Software Engineering** • Energy-efficient programming techniques • Software lifecycle and sustainability considerations • Tools for measuring and optimizing software energy consumption 3. **Sustainability in AI and Machine Learning** • Carbon footprint of AI models: challenges and trade-offs • Strategies for reducing energy consumption in training and inference • Green AI techniques: model compression, sparsity, federated learning 4. **Process Modeling for Sustainable IT** • Process modeling and business process optimization for sustainability • Green Business Process Management (BPM) • Case studies of IT-driven sustainability initiatives 5. **Regulations, Standards, and Ethical Considerations** • Legal frameworks and policies (EU AI Act, GDPR, ISO standards) • Corporate responsibility and IT governance in sustainability • Ethical considerations in Green IT and Green AI 6. **Future Trends and Research Directions** • Advances in energy-efficient computing and AI • Innovations in sustainable IT hardware and cloud infrastructures • The role of AI in supporting global sustainability efforts
Literatur	1. Hilty, L. M., & Aebischer, B., ICT Innovations for Sustainability. Springer. 2. Strubell, E., Ganesh, A., & McCallum, A. (2019), Energy and Policy Considerations for Deep Learning in NLP, https://arxiv.org/abs/1906.02243 3 . Bolón-Canedo, V., et al. (2023), Green Machine Learning: Advances in Energy-Efficient AI Models, Proceedings of ESANN 2023. 4. European Parliament, AI Act: Environmental Regulations for Artificial Intelligence in the EU Further literature will be provided during the course.

Transferkolloquium (1513100)

Transfer colloquium

Art des Moduls Pflichtmodul	**Sprache** Deutsch	**SWS** 4	**ECTS** 5
Häufigkeit Jedes Semester	**Dauer** 4 Semester	**Studiensemester** 7	**Lehr- und Lernformen** Seminar
Modulverantwortung	Prof. Dr.-Ing. Sebastian Biedermann		
Dozierende	Prof. Dr. Peter Braun, Prof. Dr.-Ing. Sebastian Biedermann		
Verwendbarkeit	Bachelor Informatik		
Aufwand	*Gesamt* 150	*Präsenzzeit* 15	*Selbststudium* 135
Voraussetzungen	*nach SPO*: keine *empfohlen*: keine		
Prüfung	*Art der Prüfung*: Portfolio *Art der Note*: ME/OE		
Lernergebnisse	• Reflexionsfähigkeit: Die Studierenden sind in der Lage, ihre Praxiserfahrungen systematisch zu reflektieren und daraus Schlüsse für ihre berufliche Entwicklung zu ziehen. • Problemlösungskompetenz: Die Studierenden entwickeln Strategien zur Lösung von Problemen, die bei der Abstimmung zwischen Studium und Praxis auftreten. • Anwendungsorientierung: Die Studierenden können theoretisch erworbenes Wissen praxisnah im Unternehmen anwenden und somit den Wissenstransfer zwischen Hochschule und Praxis optimieren. • Kommunikationsfähigkeit: Die Studierenden sind in der Lage, ihre Praxiserfahrungen klar und strukturiert zu präsentieren und effektiv zu kommunizieren. • IT-Sicherheitsbewusstsein: Die Studierenden verstehen die Bedeutung von IT-Sicherheit im Unternehmenskontext und können entsprechende Maßnahmen in ihrer täglichen Arbeit umsetzen. • Projektdokumentation: Die Studierenden sind in der Lage, Projektdokumentationen professionell zu erstellen und zu präsentieren, um ihre Arbeit transparent und nachvollziehbar zu machen. • Kontinuierliche Verbesserung: Die Studierenden lernen, Feedback konstruktiv zu nutzen, um ihre Arbeitsweise und die Zusammenarbeit zwischen Hochschule und Unternehmen kontinuierlich zu verbessern. • Karriereplanung: Die Studierenden entwickeln ein Verständnis für ihre beruflichen Entwicklungsmöglichkeiten und können fundierte Entscheidungen für ihre Karriereplanung treffen.		
Modulinhalte	• Einführung und Überblick: Vorstellung des Transferkolloquiums und seiner Ziele, Erläuterung der Bedeutung des Erfahrungsaustauschs zwischen Hochschule und Unternehmen für duale Studierende. • Reflexion der Praxiserfahrungen: Tiefergehende Reflexion der im Studium und in der Praxis erworbenen Kenntnisse und Fertigkeiten sowie deren Anwendung im betrieblichen Alltag. • Erfahrungsaustausch: Moderierte Diskussionen über Herausforderungen und Lösungen bei der Abstimmung zwischen Hochschulstudium und praktischer Arbeit im Unternehmen. • Best-Practice-Beispiele: Vorstellung und Diskussion erfolgreicher Projekte und Praktiken aus dem Unternehmen, die von den Studierenden im Studium und in der Praxis umgesetzt wurden. • Problemlösungsstrategien: Entwicklung und Präsentation von Lösungsansätzen für typische Probleme, die im dualen Studium auftreten können, einschließlich Zeitmanagement und Prioritätensetzung. • Anwendung theoretischer Kenntnisse: Vertiefung des Verständnisses, wie theoretisches Wissen aus den Hochschulmodulen praktisch im Unternehmen eingesetzt werden kann.		

	• IT-Sicherheit im Unternehmen: Einführung in die Bedeutung und Umsetzung von IT-Sicherheitsmaßnahmen in der Praxis, basierend auf theoretischen Grundlagen und praktischen Anwendungen. • Projektdokumentation und -präsentation: Schulung in der Erstellung und Präsentation von Projektdokumentationen, um die im Unternehmen durchgeführten Arbeiten strukturiert darzustellen. • Feedback und kontinuierliche Verbesserung: Systematische Sammlung und Analyse von Feedback aus den Unternehmen und der Hochschule zur kontinuierlichen Verbesserung des dualen Studiums. • Berufliche Weiterentwicklung und Karriereplanung: Diskussion von Karrierewegen und Entwicklungsmöglichkeiten für duale Studierende, basierend auf den im Studium und in der Praxis erworbenen Kompetenzen.
Literatur	Wird im Seminar bekannt gegeben.

Modulverzeichnis