MODULHANDBUCH

Professional Software Engineering, MSc.

Masterprogramm (gem. §33 LHG BaWü)

Hochschule Reutlingen
Fakultät Informatik
Alteburgstraße 150
72762 Reutlingen
https://www.inf.reutlingen-university.de/home/

Inhaltsverzeichnis

Kurzbeschreibung und Ziele des Studienprogramms

Agile Methoden, DevOps, Microservices und Cloud-Computing haben das Berufsbild von Softwareentwickler*innen verändert. Es braucht spezielle Kenntnisse, Fertigkeiten und Kompetenzen, um Anwendungen für die Cloud zu entwickeln oder dorthin zu migrieren. Um die Vorteile der Cloud wirklich ausnützen zu können, müssen Anwendungen so programmiert werden, dass sie kontinuierlich überarbeitet, getestet, gebaut, in der Cloud automatisiert bereitgestellt und überwacht werden können. Die strikte Trennung von Entwicklung und Betrieb gibt es dabei nicht mehr - “you build it, you run it” (DevOps). Ermöglicht werden dadurch neue Softwarearchitekturen, wie Microservices oder Serverless Computing, welche ihrerseits neue Anforderungen an den Entwurf und die Entwicklung von Anwendungen mit sich bringen.

Absolventinnen und Absolventen des Studienprogramms sind Softwareentwickler/-innen, die Anwendungen mit modernen Methoden und Werkzeugen, insbesondere für moderne Cloud-Umgebungen, entwickeln können.

Neben den dazu notwendigen technischen und methodischen Kenntnissen, Fertigkeiten und Kompetenzen sollen den Absolventinnen und Absolventen die folgenden Werte und Prinzipien vermittelt werden:

- Wir arbeiten agil und stellen den Kundenutzen in den Mittelpunkt unserer Arbeit.
- Wir legen Wert auf Qualität im Software Engineering und wenden entsprechende Prinzipien an.
- Wir verpflichten uns zu verantwortungsvollem, ethischem Handeln und respektieren das Recht auf Datenschutz der Nutzer.

Eine Übersicht und die detaillierte Beschreibung der Module finden sich in den folgenden Abschnitten.

Modulübersicht Professional Software Engineering (MSc.)

<table>
<tr><th>Semester</th><th colspan="30">Abschluss Master of Science</th></tr>
<tr><td>4</td><td colspan="30">Master-Thesis</td></tr>
<tr><td>3</td><td colspan="5">Wahlpflichtmodul 1</td><td colspan="5">Wahlpflichtmodul 2</td><td colspan="10">Softwareprojekt 2</td><td colspan="10" rowspan="3">Forschungsarbeit, Praxisprojekt oder Berufspraxis (Angleichungsleistung bei Vorstudium < 210 ECTS)</td></tr>
<tr><td>2</td><td colspan="5">Frontend-Entwicklung</td><td colspan="5">Backend-Entwicklung</td><td colspan="5">Softwarearchitektur</td><td colspan="5">Softwareprojekt 1</td></tr>
<tr><td>1</td><td colspan="5">Methoden und Technologien professioneller Programmierung</td><td colspan="5">Software Engineering</td><td colspan="5">Datenbanksysteme</td><td colspan="5">Cloud Computing</td></tr>
<tr><td>ECTS</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td></tr>
</table>

1 ECTS bedeutet ca. 30 Stunden Aufwand (Präsenz und Eigenleistung)

M 1 Methoden und Technologien professioneller Programmierung

Modul:	Methoden und Technologien professioneller Programmierung
Kürzel:	M 1
Untertitel:	
Lehrveranstaltungen:	Vorlesung
Modulverantwortlicher:	Prof. Dr. Christian Kücherer
Dozent(in):	Prof. Dr. Christian Kücherer Prof. Dr. Peter Hertkorn Herr Paul Lajer
Sprache:	Deutsch
Zuordnung zum Curriculum:	Professional Software Engineering, Master, Pflichtfach, 1. Semester
Lehrform / SWS:	Vorlesung, Blockveranstaltung / 4 SWS
Arbeitsaufwand:	Präsenzstudium 60 Stunden Eigenstudium 90 Stunden
Kreditpunkte:	5 ECTS
Voraussetzungen nach StuPro:	Keine
Empfohlene Voraussetzung:	Keine
Studien-/Prüfungsleistungen/ Prüfungsform:	Projektarbeit

Modulziele:

Diese Veranstaltung hebt alle Teilnehmerinnen und Teilnehmer des Masterstudiengangs auf einen gemeinsamen Kenntnis- und Wissensstand hinsichtlich Ihrer Programmierkenntnisse und die für moderne Software-Entwicklung notwendige Werkzeugkette. In der Veranstaltung werden die Anknüpfungspunkte zur ebenfalls im ersten Semester stattfindenden Veranstaltung Software Engineering aufgezeigt. Insbesondere lernen Teilnehmerinnen und Teilnehmer, worauf es bei der Entwicklung von Quellcode ankommt und wie die Vielzahl an unterschiedlichen Werkzeugen über den gesamten Entwicklungszyklus verwendet werden können. Dabei geht es neben der Entwicklung von wartbarem und verständlichem Code mit den Prinzipien von Clean Code und ständigem Refactoring auch um den Einsatz von Testframeworks zur Sicherstellung von Softwarequalität bis hin zur kompletten Infrastruktur für die kontinuierliche Auslieferung von Software mit Continuous Integration und Continuous Deployment (CI/CD).

Angestrebte Lernergebnisse

Kenntnisse:

Teilnehmerinnen und Teilnehmer in diesem Modul lernen, wie aktuelle Infrastrukturen und die dabei verwendeten Werkzeuge effektiv und effizient in aktuellen Software-Entwicklungsprojekten verwendet werden. Das Hauptaugenmerk liegt dabei zum einen auf dem methodisch-konzeptionellen Aspekt, der den Werkzeugen zugrunde liegt, und zum anderen auf der praktischen Anwendung der Werkzeuge. Die zentralen Kenntnisse, die dabei vermittelt werden sind: Prinzipien für objektorientierte Entwicklung, Entwicklung von Quellcode hoher Qualität, Versionierungsstrategien, Einsatz von Testwerkzeugen auf unterschiedlichen Abstraktionsebenen, Continuous Delivery und Continuous Deployment, Einsatz von Software-Metriken und statischer Codeanalyse, Code-Dokumentation, kollaborative Softwareentwicklung sowie Entwicklungsumgebungen. Bei der Versionierung und Konfiguration von Quellcode werden verschiedene Branching-Strategien, Feature Toggles, Tagging und Merging-Strategien betrachtet.

Fertigkeiten:

Die Teilnehmerinnen und Teilnehmern

- Erstellen und verwalten mit Hilfe von Anforderungsmanagement-Werkzeugen die Anforderungen an ein Softwaresystem und überführen diese mit agiler Vorgehensweise schrittweise in ein lauffähiges System.
- Erstellen unter Berücksichtigung der Prinzipien von klassischen Design Patterns selbstständig komponentenbasierte Softwaresysteme.
- Wenden die Prinzipien von Clean Code und Clean Architecture an und erstellen dadurch wartbaren Code.
- Verwalten erstellten Code mit Hilfe von Versionierungssystemen und entwickeln den Code in Teams gemeinsam weiter.
- Prüfen erstellten Code durch Testframeworks funktional und nichtfunktional.
- Dokumentieren erstellten Code.
- Wenden die Grundlagen von Architekturmustern praktisch in Projekten an.
- Liefern Softwaresysteme kontinuierlich als lauffähige Anwendungen mit Hilfe von Continuous Integration, Continuous Delivery und Continuous Deployment aus und verwenden dabei die dafür notwendige Infrastruktur.

Kompetenzen:

Nach Abschluss des Moduls sind die Teilnehmerinnen und Teilnehmern in der Lage:

LE#	Lernergebnis (LE)	Geprüft durch
LE1	Unter Berücksichtigung von Design Patterns, objektorientiertem Design und den SOLID Prinzipien stabilen, hochqualitativen und wartbaren Quellcode zu erstellen.	Testataufgabe
LE2	Die Basis-Architektur- von Softwaresystemen zu erstellen und mit geeigneten Patterns Software zu entwickeln.	Testataufgabe
LE3	Die Prinzipien von Clean Code und Clean Architecture anzuwenden und Software sinnvoll zu dokumentieren	Testataufgabe
LE4	Die Methoden von CI / CD praktisch anzuwenden, zur Problemstellung passende Werkzeuge auszuwählen sowie die gesamte Deployment-Pipeline zu dokumentieren.	Hausarbeit und Referat
LE5	Qualitätssicherung mit Testframeworks und statischer Analyse durch Software-Metriken durchzuführen.	Testataufgabe
LE6	Entwicklungsumgebungen effektiv zu nutzen und die Probleme bei kollaborativer Software-Entwicklung durch geeigneten Werkzeug- und Methodeneinsatz zu lösen.	Testataufgabe
LE7	Versionierungsstrategien und Werkzeuge sinnvoll in Projekten anzuwenden.	Referat

Inhalt:

Zu Beginn der Veranstaltung befassen sich die Teilnehmerinnen und Teilnehmern mit den Grundlagen der modernen Software-Entwicklung und wenden Software Design Patterns an, um ihre objektorientierten Systeme zu strukturieren und anhand von etablierten Standards mit Funktionalität zu versehen (LE1). Dadurch entsteht stabiler, hochqualitativer und wartbarer Quellcode, der die Evolution des Systems vereinfacht. Ausgehend von definierten Anforderungen erstellen Teilnehmerinnen und Teilnehmer die Basis-Architektur eines Softwaresystems und setzen dabei geeignete Patterns ein (LE2). Als weiterer Theoriebaustein werden die Prinzipien von Clean Code und Clean Architecture vermittelt, die Teilnehmerinnen und Teilnehmer an einem praktischen Beispiel anwenden. Dabei geht es nicht nur um die Verbesserung der Qualität von Quellcode und der strukturgebenden Architektur, sondern auch darum, den Code der Software sinnvoll zu dokumentieren (LE3). Stand der Technik ist die kontinuierliche Kompilierung und Integration der Komponenten eines Softwaresystems, um einer schleichenden Verschlechterung des Codes vorzubeugen. Dazu werden Werkzeugketten, die diese Deployment-Pipeline realisieren, durch die Teilnehmerinnen und Teilnehmern ausgewählt zu einem sinnvollen Ganzen zusammengesetzt (LE4). Um eine kontinuierliche Funktionalität des Gesamtsystems herzustellen, werden Regressionstests verwendet, um die Seiteneffektfreiheit von Änderungen zu prüfen. Dazu werden automatisierte Testframeworks eingesetzt, die Teilnehmerinnen und Teilnehmer am praktischen Beispiel auswählen und einsetzen (LE5). Mit Hilfe

von Kollaborations- und Versionsverwaltungswerkzeugen lernen Teilnehmerinnen und Teilnehmer, die Probleme bei verteilter Software-Entwicklung zu erkennen und geeignete Maßnahmen zu ergreifen (L6). Schließlich werden die Grundlagen verschiedener Versionierungsansätze und Werkzeuge vermittelt, so dass eine parallele Entwicklung an der gleichen Code-Basis durch viele EntwicklerInnen ermöglicht wird (LE7).

Medienformen:

Das Lehrmaterial besteht aus einem Folienskript, das in elektronischer Form vorliegt, Übungsblättern sowie Programmbeispielen. Seminaristischer Unterricht mit Whiteboard, PC-Beamer und Präsentationsfolien, bei dem Beispiele zu den theoretischen Inhalten veranschaulicht werden sowie Demonstration von Beispielprogrammen und interaktiver Programmentwicklung. Die Teilnehmerinnen und Teilnehmern bearbeiten individuell oder in Gruppen Übungsaufgaben zum Themengebiet der modernen Software-Entwicklung mit Betreuung durch DozentInnen. Ein weiteres Element ist die Erarbeitung durch Teilnehmerinnen und Teilnehmer als Inverted Classroom: Teilnehmerinnen und Teilnehmer müssen ein Teilgebiet der kompletten Werkzeug-Infrastruktur anhand einer definierten Problemstellung ausarbeiten. Durch ein Referat mit 90-minütigen Workshop wird das gesammelte Wissen den anderen Teilnehmenden vermittelt.

Literatur:

- Alur, Deepak; Crupi, John; Malks, Dan (2003): Core J2EE patterns: best practices and design strategies, 2. Auflage, Upper Saddle River, NJ: Prentice Hall PTR; Palo Alto, Calif., Sun Microsystems Press.
- Duvall, Paul M.; Matyas, Steve und Glover, Andrew (2010): Continuous integration: improving software quality and reducing risk, 5. Auflage, Upper Saddle River, NJ; Munich [u.a.] : Addison-Wesley
- Fowler, Martin (2000): Refactoring: wie Sie das Design vorhandener Software verbessern. München [u.a.], Addison Wesley,
- Fowler, Martin (2008): Patterns of enterprise application architecture, 14. Auflage. Boston, Mass.; Munich [u.a.], Addison-Wesley
- Gamma, Erich (2005): Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software, München [u.a.], Addison-Wesley
- Humble, Jez and Farley, David (2011): Continuous delivery: reliable software releases through build, test, and deployment automation, 1. Auflage, Upper Saddle River, NJ ; Munich: Addison-Wesley
- Link, Johannes; Fröhlich, Peter (2002): Unit Tests mit Java: der Test-first-Ansatz, 1. Auflage, Heidelberg, dpunkt
- Loeliger, Jon (2009): Version Control with Git: Powerful tools and techniques for collaborative software development, O'Reilly Media.
- Martin, Robert C. (2009): Clean code: Refactoring, Patterns, Testen und Techniken für sauberen Code, 1. Auflage, Heidelberg; München; Landsberg; Frechen; Hamburg, mitp
- Martin, Robert C. (2014): Clean Coder: Verhaltensregeln für professionelle Programmierer, 1. Auflage Heidelberg, mitp.
- Tabaka, Jean (2006): Collaboration Explained: Facilitation Skills for Collaborative Leaders. Agile Software Development Series, Addison Wesley.

M 2 Software Engineering

Modul:	Software Engineering
Kürzel:	M 2
Untertitel:	
Lehrveranstaltungen:	Vorlesung
Modulverantwortlicher:	Prof. Dr. Christian Decker
Dozent(in):	Prof. Dr. Christian Decker Prof. Dr. Christian Kücherer Dominik Neumann
Sprache:	Deutsch
Zuordnung zum Curriculum:	Professional Software Engineering Master, Pflichtfach, 1. Semester
Lehrform / SWS:	Vorlesung / 4 SWS
Arbeitsaufwand:	Präsenzstudium 60 Stunden Eigenstudium 90 Stunden
Kreditpunkte:	5 ECTS
Voraussetzungen nach StuPro:	Keine
Empfohlene Voraussetzung:	Keine
Studien-/Prüfungsleistungen/ Prüfungsform:	Hausarbeit, Referat

Modulziele:

Zielgerichtete und systematische Steuerung der Softwareentwicklung bestimmt maßgeblich den Erfolg eines Softwareprojekts.

Das Modul führt die ingenieurwissenschaftliche Vorgehensweise des Software Engineerings entlang des Softwarelebenszyklus, den zugehörigen Prozessen und Vorgehensmodellen ein. Ein Schwerpunkt sind agile Vorgehensmodelle. Eine zentrale Rolle im Modul spielt das Requirements Engineering und die Zusammenhänge zu den anderen Phasen des Softwarelebenszyklus. Anforderungen, die bei und mit Stakeholdern systematisch erhoben und dokumentiert werden, bilden die Grundlage für den Umfang und Funktion des zu schaffenden Systems. Die dabei entstehende Spezifikation beschreibt die zu erwartende Funktionalität für alle Formen der Qualitätssicherung, wie z.B. Testen und Testprozesse. Weitere Formen der Qualitätssicherung umfassen organisatorische Maßnahmen, wie z.B. die Einführung regelmäßiger Reviews oder die Implementierung des V-Modell des Testens. Davon beeinflusst werden das Systemdesign und die Aktivitäten der Softwaremodellierung.

Schließlich vermittelt das Modul die Grundlagen des strategischen Domain Driven Designs (DDD), um ein wichtiges Fundament für die Herausbildung und das Verständnis moderner und in der Praxis häufig verwendeter Ansätze von Softwarearchitekturen innovativer Softwareprodukte zu schaffen. Die Anwendung dieser Herangehensweise wird den Modulteilnehmern durch eine praktische Simulation eines kundenahen Umfelds nähergebracht. Mit dem strategischen Domain Driven Design grenzt sich das Modul Software Engineering auch vom taktischen Domain Driven Design des Moduls Softwarearchitektur ab.

Angestrebte Lernergebnisse

Kenntnisse:

Die Teilnehmer erwerben Kenntnisse und Erfahrungen über ingenieurwissenschaftliche Methoden bei der Erstellung und Durchführung von Softwareprojekten und Wissen über unterschiedlichen systematischen Vorgehensweisen bei der Entwicklung von Software. Sie wissen um wichtige aktuelle Standards und Vorgehensweisen zur Festlegung der Anforderungen an ein System wie z.B. gängige Erhebungstechniken und Dokumentationsformen der Softwarespezifikation. Für die Umsetzung vermittelt das Modul an die Teilnehmer das Know-How im strategischen DDD und Systemdesign. Sie lernen die Methoden des Event Storming, Context Mapping und Ubiquitous Language kennen. Schließlich werden die Teilnehmer in den Praktiken der durchgängigen Qualitätssicherung ausgebildet. Das umfasst die Softwarequalitätssicherung durch testgetriebene Methoden und organisatorischen Methoden wie das V-Modell des Testens und schließt Dokumentationsartefakte vollständig ein.

Fertigkeiten:

Die Teilnehmer erwerben in diesem Modul die Fertigkeiten eines Softwareingenieurs und können diese anwenden. Das heißt im Einzelnen: Passende Vorgehensmodelle auszuwählen, ggf. anzupassen und danach zielgerichtet in der Softwareentwicklung vorzugehen; Software in einem systematischen und nachvollziehbaren Prozess zu spezifizieren, zu verschriftlichen und die Qualitätssicherung zu gewährleisten; ein passendes Systemdesign mit Hilfe der Praktiken und Methoden des strategischen DDD für kundenspezifische Softwareprojekte zu entwerfen und umsetzbar zu gestalten.

Kompetenzen:

Die Teilnehmer sind in der Lage erfolgreich komplexe Softwareprojekte zu spezifizieren und zu entwerfen. Sie qualifizieren sich, Verantwortung für Softwareentwicklungen zu übernehmen und zu erfolgreichen sowie umsetzbaren Softwareprojekten weiterzuführen. In dem Modul erlangen die Teilnehmer eine kompetente Grundlagenausbildung für Entwicklung moderner und in der Praxis häufig verwendeten Ansätze von Softwarelösungen in innovativen Softwareprodukten beispielsweise mit Cloud-native Anwendungsdesigns und Verwendung von Microservice-Architekturen. Teilnehmer und Teilnehmerinnen erlangen ein Verständnis für das Spannungsfeld der Ethik im Softwareengineering und kennen Vorgehen, um ethische Prinzipien anzuwenden.

LE#	Lernergebnis (LE)	Geprüft durch
LE1	Prozesse und Vorgehensmodelle der plangetriebenen und agilen Softwareentwicklung	Hausarbeit
LE2	Anforderungen und Requirements Engineering Prozess, Erhebungstechniken und Dokumentation, SW Qualitätssicherung, Organisatorische Qualitätssicherung und statische Prüfung, z.B. Reviews	Hausarbeit
LE3	Use Cases, Systemdesign und Definition von Systemgrenzen; Testspezifikationen, testgetriebene Entwicklung (TDD); V Modell des Testens	Hausarbeit
LE4	Strategisches Domain Driven Design (DDD), Identifikation von fachlichen Domänen-Schnitten (Bounded Context); Metoden u. Praktiken: Event Storming, Context Mapping, Ubiquitous Language	Referat
LE5	Simulation von DDD Praktiken in einem kundenahen Umfeld	Referat

Inhalt:

- Prozesse und Vorgehensmodelle des Softwareengineerings
- Agile vs. Plangetriebene Prozesse
- Anforderungen und Requirements Engineering Prozess
- Software-Qualitätssicherung, organisatorische Qualitätssicherung und statische Prüfung, z.B. Reviews
- Testspezifikationen, Test-Driven-Development (TDD), V Modell des Testens
- Erhebungstechniken und Qualitätsanforderungen spezifizieren
- Dokumentationsstandards und deren Qualitätssicherung
- Spezifikation von Use Cases und Systemgrenzen
- Strategisches Domain Driven Design (DDD) und Identifizieren von Bounded Context
- Methoden und Techniken des strategischen DDD: Event Storming, Context Mapping, Ubiquitous Language
- Simulation von Praktiken des strategischen DDD in einem kundennahen Umfeld
- Ethik im Softwareengineering anhand aktueller Ethikrichtlinien für die Softwareentwicklung

Übungen werden in wechselnden Teamzusammenstellungen. Ziel ist es, die Inhalte zu vertiefen und verschiedene Sichtweisen zu erschließen.

Medienformen:

Vorlesung, Übungsaufgaben, Skript mit den PDF der Folien der Vorlesung, beispielhafte Publikationen, Simulation eines kundennahen Umfelds

Literatur:

- Sommerville, Ian. Software Engineering, Pearson Studium; Auflage: 9. Aktual. (1. März 2012), ISBN-10: 3868940995
- Patton, Jeff. User Story Mapping- Nutzerbedürfnisse besser verstehen als Schlüssel für erfolgreiche Produkte , O'Reilly; Auflage: 1 (30. April 2015) ISBN-10: 3958750672
- Ludewig, J., Lichter H.: Software Engineering – Grundlagen, Menschen, Prozesse, Techniken. dpunkt.verlag Heidelberg, 2007. ISBN 3-89864-268-2
- Basiswissen Requirements Engineering: IREB Foundation Level, Klaus Pohl (Autor), Chris Rupp (Autor)
- Lauesen S: Software Requirements: Styles & Techniques: Styles and Techniques, 2002, Addison-Wesley, ISBN 978-0201745702
- Spillner A, Linz T: Basiswissen Softwaretest: Aus- und Weiterbildung zum Certified Tester - Foundation Level nach ISTQB-Standard, 5. Aufl. 2012, dpunkt.verlag GmbH; ISBN 978-3864900242
- Vernon, Vaughn, Domain-Driven Design kompakt, (deutsche Übersetzung von C. Lilienthal), dpunkt Verlag 2017, ISBN: 978-3864904394

Weitere Literatur wird während der Vorlesung bekannt gegeben.

M 3 Datenbanksysteme

Modul:	Datenbanksysteme
Kürzel:	M 3
Untertitel:	
Lehrveranstaltungen:	Vorlesung
Modulverantwortlicher:	Prof. Dr. Peter Hertkorn
Dozent(in):	Prof. Dr. Peter Hertkorn
Sprache:	Deutsch
Zuordnung zum Curriculum:	Professional Software Engineering Master, Pflichtfach, 1. Semester
Lehrform / SWS:	Vorlesung, Blockveranstaltung / 4 SWS
Arbeitsaufwand:	Präsenzstudium 60 Stunden Eigenstudium 90 Stunden
Kreditpunkte:	5 ECTS
Voraussetzungen nach StuPro:	Keine
Empfohlene Voraussetzung:	Keine
Studien-/Prüfungsleistungen/ Prüfungsform:	Projektarbeit

Modulziele:

Die Teilnehmerinnen und Teilnehmern erlangen Kenntnisse über die Funktionsweise von modernen Datenbanksystemen und unterschiedlichen Datenbanktechnologien. Sie verstehen die zugrundeliegenden Prinzipien, Methoden und Techniken und können die theoretischen Kenntnisse in der Praxis anwenden. Im weiteren Verlauf des Studiums soll mit dem erfolgreichen Bestehen des Moduls gewährleistet werden, dass die Teilnehmerinnen und Teilnehmern in der Lage sind, geeignete Datenbanktechnologien für gegebene Problemstellungen auszuwählen und die unterschiedlichen Datenbanksysteme mit Hilfe von Datenbank- und Programmiersprachen erstellen und nutzen zu können.

Angestrebte Lernergebnisse

Kenntnisse:

Die Teilnehmerinnen und Teilnehmer

- kennen Methoden zum Zugriff auf eine Datenbank aus einem Anwendungsprogramm heraus.
- sind in der Lage, Methoden zur Migration und Versionierung von Daten zu beschreiben.
- können Eigenschaften objektorientierter und objektrelationaler Datenbanken erklären.
- kennen Methoden zur Speicherung semistrukturierter Daten.
- können Konzepte neuerer Entwicklungen von Datenbanksystemen beschreiben und die Unterschiede zu relationalen Datenbanksystemen erläutern.
- sind in der Lage, die Eigenschaften verteilter Systeme im Hinblick auf Datenbanksysteme zu erklären.
- kennen die Eigenschaften von dokumentorientierten Datenbanken und können diese beschreiben und bewerten.
- können die Eigenschaften von Key-Value-Datenbanken erläutern.
- sind in der Lage, die Eigenschaften von Graphdatenbanken zu erklären.
- kennen die Eigenschaften von spaltenorientierten Datenbanken.
- können verschiedene Methoden zur Skalierung von Datenbanksystemen beschreiben und bewerten.

Fertigkeiten:

Die Teilnehmerinnen und Teilnehmer formulieren Abfragen sowie Änderungen an relationale Datenbanken aus einem Anwendungsprogramm. Sie wenden Methoden zur Migration und Versionierung von Daten an. Die Teilnehmerinnen und Teilnehmer erstellen Datenbankschemas und Abfragen für objektrelationale Datenbanken sowie Abfragen für semistrukturierte Daten anhand der XML-Erweiterungen des relationalen Modells. Sie analysieren die Anforderungen für gegebene Problemstellungen und wählen eine geeignete Form von NoSQL-Datenbank aus. Die Teilnehmerinnen und Teilnehmer erstellen Datenmodelle für die verschiedenen Arten von NoSQL-Datenbanken und erstellen Abfragen sowie Änderungen mittels geeigneter Datenbanksprachen an die NoSQL-Datenbanken. Die Teilnehmerinnen und Teilnehmer wenden Methoden zur Skalierung von Datenbanksystemen an.

Kompetenzen:

Nach Abschluss des Moduls sind die Teilnehmerinnen und Teilnehmer in der Lage:

LE#	Lernergebnis (LE)	Geprüft durch
LE1	Methoden zum Zugriff auf eine Datenbank aus einem Anwendungsprogramm anzuwenden.	Artefakt
LE2	Methoden zur Migration und Versionierung von Daten anzuwenden.	Artefakt
LE3	Modellierungsalternativen bei der Erstellung der Datenbanken zu bewerten.	Artefakt
LE4	Datenbanken für unterschiedliche Datenmodelle mit Datenbanksprachen zu erstellen.	Artefakt
LE5	Für gegebene Anforderungen Abfragen an die Datenbank zu formulieren und alternative Möglichkeiten bei Abfragen an die Datenbank zu bewerten und hinsichtlich Performanz zu beurteilen.	Artefakt

LE6	Unterschiedlichen Datenbanktechnologien für einen gegebenen Anwendungsfall bewerten und eine geeignete Datenbanktechnologie auswählen zu können.	Artefakt
LE7	Aktuelle Entwicklungen im Bereich Datenbanksysteme zu beurteilen und sich anzueignen.	Artefakt

Inhalt:

In der Vorlesung werden die Kenntnisse des relationalen Modells vertieft (LE1) und um den Einsatz von Methoden zur Migration und Versionierung von Daten erweitert (LE2). Neben klassischen relationalen Datenbanksystemen werden sowohl objektrelationale Datenbanken und die XML-Erweiterungen des relationalen als auch die verschiedenen Arten von NoSQL-Datenbanken ausführlich behandelt und deren Eigenschaften mit denen der relationalen Datenbanksysteme verglichen (LE3-6). Der Zugriff auf Datenbanken aus einer Anwendung heraus wird vorgestellt und an Beispielprogrammen erläutert (LE1). Des Weiteren werden neuere Entwicklungen im Bereich Datenbanksysteme vorgestellt und deren Eigenschaften mit denen der behandelten Datenbanksysteme verglichen (LE7). Bei der praktischen Umsetzung wird darauf geachtet, dass in der Industrie genutzte Werkzeuge eingesetzt werden, so dass auch ein praktisches Wissen erworben wird.

Medienformen:

Das Lehrmaterial besteht aus einem Folienskript, das in elektronischer Form vorliegt, Übungsblättern sowie Programmbeispielen. Seminaristischer Unterricht mit Whiteboard, PC-Beamer und Präsentationsfolien, bei dem Beispiele zu den theoretischen Inhalten veranschaulicht werden sowie Demonstration von Beispielprogrammen und interaktiver Programmentwicklung. Die Teilnehmerinnen und Teilnehmer bearbeiten individuell oder in Gruppen Übungsaufgaben zum Themengebiet Datenbanksysteme mit Betreuung durch den Dozenten.

Literatur:

- Ambler, Scott W. und Pramodkumar J. Sadalage (2011): Refactoring Databases: Evolutionary Database Design. Addison-Wesley.
- Edlich, Stefan und Achim Friedland (2011): NoSQL: Einstieg in die Welt nichtrelationaler Web 2.0 Datenbanken. 2. Auflage. Hanser.
- Fowler, Adam (2015). NoSQL for Dummies. Dummies Tech.
- Harrison, Guy (2015): Next Generation Databases: NoSQL, NewSQL, and Big Data. Apress.
- McCreary, Dan und Ann Kelly (2013): Making Sense of NoSQL: A Guide for Managers and the Rest of Us. Manning.
- Perkins, Luc et al. (2018): Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement. 2. Auflage. O'Reilly.
- Robinson, Ian et al. (2015): Graph Databases: New Opportunities for Connected Data. 2. Auflage. O'Reilly.
- Sadalage, Pramod J. und Martin Fowler (2012): NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence. Addison-Wesley.
- Sullivan, Dan (2015): NoSQL for Mere Mortals. Addison-Wesley.
- Trelle, Tobias (2014): MongoDB: Der praktische Einstieg. Dpunkt.
- Vonhoegen, Helmut (2018): XML: Einstieg, Praxis, Referenz. 9. Auflage. Rheinwerk Computing.

M 4 Cloud Computing

Modul:	Cloud Computing
Kürzel	M 4
Lehrveranstaltungen:	Vorlesung
Modulverantwortlicher:	Prof. Dr. Marcus Schöller
Dozent(in):	Prof. Dr. Marcus Schöller Prof. Dr. Wolfgang Blochinger André Lindenberg
Sprache:	Deutsch / Englisch
Zuordnung zum Curriculum:	Professionelle Software-Entwicklung Master, Pflichtfach, 1. Semester
Lehrform/SWS:	Vorlesung / 4 SWS
Arbeitsaufwand:	Präsenzstudium 60 Stunden Eigenstudium 90 Stunden
Kreditpunkte:	5 ECTS
Voraussetzungen nach StuPro:	keine
Empfohlene Voraussetzung:	
Studien-/Prüfungsleistungen/ Prüfungsform:	Projektarbeit

Modulziele:

Cloud Computing ist mittlerweile ein zentraler Bestandteil der Unternehmens-IT und ermöglicht zudem eine Vielzahl neuartiger digitaler Geschäftsmodelle und digitaler Produkte. In diesem Modul sollen Teilnehmerinnen und Teilnehmer mit wesentlichen Aspekten des Cloud Computing vertraut gemacht werden. Sie sollen insbesondere umfassende Kenntnisse über den Entwurf, die Entwicklung und den Betrieb verteilter Cloud-Dienste und -Anwendungen erwerben.

Angestrebte Lernergebnisse:

Kenntnisse:
Teilnehmerinnen und Teilnehmer dieses Moduls verfügen nach dem erfolgreichen Abschluss über Kenntnisse der wesentlichen Prinzipien und Charakteristiken des Cloud Computing. Sie kennen die grundlegenden Ausprägungen von Cloud-Diensten und ihrer Liefermodelle und haben ein Verständnis für technische, organisatorische, kommerzielle und sicherheitsrelevante Aspekte des Cloud Computing entwickelt. Sie haben vertiefte Kenntnisse über Cloud Architektur Patterns, Programmierwerkzeuge und -plattformen sowie deren Einsatzgebiete.

Die Teilnehmerinnen und Teilnehmer haben ein gutes Verständnis über die fundamentalen Gesetze, Verordnungen und Strategien im Datenschutz.

Fertigkeiten:
Die Teilnehmerinnen und Teilnehmer können typische Dienst- und Liefermodelle im Hinblick auf Fallbeispiele beurteilen. Sie können Anforderungen von (Server-)Diensten analysieren und geeignete Deployment-Varianten entwickeln und bewerten. Diese Varianten reichen von in-house Server-Lösungen über hybrid Cloud Modelle bis hin zu reinen Cloud-Lösungen. Dazu wenden sie eine Reihe von erlernten Methoden an. Basierend auf diesen Anforderungen können die Teilnehmerinnen und Teilnehmer Dienste entwerfen und entwickeln, die sich die Charakteristiken der Cloud zu Nutze machen. Des Weiteren kennen die Teilnehmerinnen und Teilnehmer Details von Cloud-Umgebungen, was Ihnen einen tiefergehenden Vergleich der unterschiedlichen Deployment-Varianten ermöglicht. Die Teilnehmerinnen und Teilnehmer können also die Lösungen ganzheitlich analysieren und bewerten und somit technisch fundierte Entscheidungen für die Diensterbringung treffen. Sie sind zudem in der Lage, Cloud-Anwendungen hinsichtlich datenschutzrechtlicher Aspekte zu analysieren, so dass sie die geeigneten (technischen) Maßnahmen bei Entwurf und Implementierung dieser Anwendungen ergreifen können.

Kompetenzen:

LE#	Lernergebnis	Geprüft durch
LE1	Verständnis der unterschiedlichen Cloud-Business-Modelle (IaaS, PaaS, SaaS) haben und anwenden können	Projektarbeit
LE2	Komponenten und deren Aufgaben in einer Cloud-Architektur in Beziehung setzen können	Projektarbeit
LE3	Operative Aspekte einer Cloud-Infrastruktur verstehen und bewerten können	Projektarbeit
LE4	Methoden der Softwareentwicklung für eine Cloud-Plattform verstehen und anwenden können	Projektarbeit
LE5	Cloud-Anwendungen hinsichtlich datenschutzrechtlicher Aspekte analysieren können	Projektarbeit

Inhalt:

Die Dienstmodelle Infrastructure as a Service (IaaS), Platform as a Service (PaaS) sowie Software as a Service (SaaS) werden sowohl aus Anbietersicht als auch Nutzersicht dargestellt. Im Vordergrund stehen hier die Themen Softwareentwurf und -entwicklung für die Cloud als auch ein grundlegendes Verständnis für Cloud-Umgebungen. Die Liefermodelle Public Cloud, Private Cloud, Hybrid Cloud, Community Cloud werden anhand von Fallbeispielen vermittelt. Ausführlich werden technische, organisatorische, kommerzielle und sicherheitsrelevante Aspekte des Cloud Computings behandelt, bewertet und diskutiert. Es werden datenschutzrechtliche Konzepte und Regelungen vorgestellt sowie organisatorische und technische Maßnahmen zum Schutz personenbezogener Daten diskutiert.

Medienformen:

Seminaristische Vorlesung, Folien und Tafelanschrieb; Case-Studies in Kleingruppen.

Literatur:

- Antonopoulos, Nick; Gillam, Lee (2010): Cloud Computing. Principles Systems and Applications. London: Springer London (SpringerLink: Bücher, 0).
- Baun, Christian; Kunze, Marcel; Nimis, Jens; Tai, Stefan (2011): Cloud Computing. Web-basierte dynamische IT-Services. Berlin, Heidelberg: Springer Berlin Heidelberg (SpringerLink : Bücher).
- Buyya, Rajkumar (2011): Cloud computing. Principles and paradigms. Hoboken, NJ: Wiley (Wiley series on parallel and distributed computing).
- Fehling, Christoph; Leymann, Frank; Retter, Ralph; Schupeck, Walter; Arbitter, Peter (2014): Cloud Computing Patterns. Fundamentals to Design, Build, and Manage Cloud Applications. Wien: Springer.

M 5 Frontend-Entwicklung

Modul:	Frontend-Entwicklung
Kürzel:	M 5
Untertitel:	
Lehrveranstaltungen:	
Modulverantwortlicher:	Prof. Dr. Natividad Martínez Madrid
Dozent(in):	Prof. Dr. Natividad Martínez Madrid Prof. Dr. Peter Hertkorn Ralf Kretzschmar-Auer Matthias Gutbrod
Sprache:	Deutsch
Zuordnung zum Curriculum:	Professional Software Engineering Master, Pflichtfach, 2. Semester
Lehrform / SWS:	Vorlesung 4 SWS
Arbeitsaufwand:	Präsenzstudium 60 Stunden Eigenstudium 90 Stunden
Kreditpunkte:	5 ECTS
Voraussetzungen nach StuPro:	Keine
Empfohlene Voraussetzung:	Methoden und Technologien professioneller Programmierung
Studien-/Prüfungsleistungen/ Prüfungsform:	Projektarbeit

Modulziele:

Das Modul adressiert das Thema der benutzerzentrierten Frontend-Entwicklung, sowohl aus der methodischen als auch aus der technischen Perspektive. Die Studierenden erlangen Kenntnisse über die grundsätzlichen Schritte zum Design eines Frontends. Sie verstehen, wie eine gute Interaktion zwischen Frontend und Backend aussieht, können Frontendarchitekturen entwerfen sowie entsprechende Technologien und Frameworks anwenden. Die Studierenden entwerfen mobile Frontends und lernen die Eigenschaften mobiler Plattformen kennen. Nach erfolgreichem Bestehen des Moduls sind die Studierenden in der Lage, ein geeignetes Frontend für gegebene Problemstellungen zu entwerfen und mit Hilfe unterschiedlicher Technologien zu entwickeln.

Angestrebte Lernergebnisse

Kenntnisse:

- Die Studierenden kennen die Grundlagen der Kommunikationspsychologie zum UX-Design: Visualisierung, Farben/Bilder/Animationen, Verhalten, Zielgruppen, Benutzerführung, Fehlertoleranz, Feedback an den Benutzer.
- Die Studierenden sind in der Lage, die Eigenschaften barrierefreier Interaktionen zu beschreiben.
- Die Studierenden kennen konzeptuelle Werkzeuge des UX-Designs: Context Navigation Maps, Screenflows, Papierprototypen.
- Die Studierenden können die Grundlagen des Responsive Designs / Mobile-First Ansatzes erläutern.
- Die Studierenden kennen die Grundlagen von komponentenbasiertem Design im Frontend und können die Methoden zur Interaktionen zwischen Frontend und Backend beschreiben.
- Die Studierenden kennen die Grundlagen zur Programmierung von Frontends auf Basis von Web-Technologien.
- Die Studierenden sind in der Lage, die Eigenschaften moderner Frameworks für die Frontend-Entwicklung zu beschreiben und zu bewerten.
- Die Studierenden kennen verschiedene Entwurfsmuster, die bei der Entwicklung von Frontends eingesetzt werden.
- Die Studierenden kennen die Eigenschaften von mobilen Geräten als Frontend-Plattformen.
- Die Studierenden können die Unterschiede zwischen nativen und webbasierten/cross-plattform mobile Frameworks erläutern und bewerten.
- Die Studierenden sind in der Lage, die Besonderheiten der Frontendentwicklung auf nativen Frameworks zu beschreiben.

Fertigkeiten:

- Die Studierenden können für eine gegebenen Aufgabe ein Konzept erarbeiten.
- Die Studierende setzen gelernte UX-Techniken in dem Prototyping von Frontends ein (Desktop/Mobilgerät).
- Die Studierenden können ein UX-Design prototypisch umsetzen: Interaktionen zwischen Frontend und Backend gestalten, Errorhandling durchführen, ggf. das Backend mocken und die Software auf einem Server in Betrieb setzen.
- Die Studierenden erstellen Frontends auf Basis von Web-Technologien.
- Die Studierenden setzen geeignete Frameworks für die Frontend-Entwicklung ein.
- Die Studierenden analysieren die Anforderungen für gegebene Problemstellungen und wählen geeignete Technologien für die Realisierung aus.
- Die Studierenden wenden Entwurfsmuster bei der Entwicklung von Frontends an.
- Die Studierenden vergleichen unterschiedliche mobile native oder cross-plattform Frameworks unter bestimmten Anforderungen.
- Die Studierenden können für gegebene Aufgaben ein mobiles Frontend auf Basis von cross-plattform und von nativen Plattformen umsetzen.

Kompetenzen:

Nach Abschluss des Moduls sind die Studierenden in der Lage:

LE#	Lernergebnis (LE)	Geprüft durch
LE1	Methoden zur Modellierung von UX-Designs anwenden zu können.	Projektarbeit
LE2	UX-basierte Prototypen von Frontends erstellen zu können.	Projektarbeit
LE3	Frontends mit Hilfe von Web-Technologien erstellen zu können.	Projektarbeit
LE4	Alternativen beim Entwurf und der Realisierung von Komponenten bewerten zu können.	Projektarbeit
LE5	Frameworks für die Frontend-Entwicklung anwenden zu können.	Projektarbeit
LE6	Frameworks für cross-plattform mobile Frontend-Entwicklung benutzen zu können.	Projektarbeit
LE7	Aspekte von nativen mobilen Frontends entwerfen zu können.	Projektarbeit

Inhalt:

In der Vorlesung werden die Grundlagen des benutzerzentrierten Designs (UX-Design) durch die Einführung in die Kommunikationspsychologie erläutert. Unterschiedliche Methoden zur benutzerzentrierten Modellierung von Frontends werden adressiert und von den Studierenden in einem konkreten Beispiel umgesetzt (LE1). Anschließend lernen die Studierenden Techniken zum Prototyping von Frontends im Zusammenspiel mit (ggf. mocked-up) Backends. Das modellierte UX-Frontend wird dann prototypisch umgesetzt (LE2). Anhand verschiedener Problemstellungen werden unterschiedliche Web-Technologien zur Entwicklung von Frontends behandelt (LE3). Dabei werden Alternativen beim Entwurf analysiert und bewertet sowie bewährte Lösungen besprochen (LE4). Die dabei erworbenen Kenntnisse wenden die Studierenden anschließend selbständig beim Lösen von Übungsaufgaben sowie bei der Erstellung von Frontends mit Hilfe geeigneter Frameworks an (LE5). Die Studierenden sind von Anfang an mit der Entwicklung von mobilen Frontends durch den Responsive-Design/Mobile First-Ansatz vertraut. Die Studenten lernen auch die Möglichkeiten der Integration von Funktionalität und Eigenschaften des nativen Frameworks des Endgeräts durch den Cross Plattform-Ansatz (z.B. React Native) kennen (LE6). Sie analysieren und bewerten die Vor- und Nachteile von webbasierten Frontends versus nativen mobilen Frontends. Anschließend werden die Grundlagen der nativen mobilen Programmierung (z.B. Android) durch die Analyse von Beispielen aus ausgewählten Themengebieten behandelt (LE7).

Medienformen:

Das Lehrmaterial besteht aus einem Folienskript, das in elektronischer Form vorliegt, Videos, Design- und Programmbeispielen und praktischen Aufgaben. Der seminaristische Unterricht mit Whiteboard, PC-Beamer und Präsentationsfolien, bei dem Beispiele zu den theoretischen Inhalten veranschaulicht werden, wird kombiniert mit Praxisteilen und Demonstration von Beispielprogrammen sowie interaktiver Design- und Programmentwicklung. Die Studierenden bearbeiten individuell oder in Gruppen Aufgaben mit Betreuung durch den Dozenten.

Literatur:

- Banks, Alex und Eve Porcello (2017): Learning React: Functional Web Development with React and Flux. O'Reilly.
- Beyer, Hugh und Karen Holtzblatt (1998): Contextual Design: Defining Customer-Centered Systems. Morgan Kaufmann Publishers.
- Constantine, Larry und Lucy Lockwood (2004): Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design. 4. Auflage. ACM Press.
- Flanagan, David (2011): JavaScript: The Definitive Guide. 6. Auflage. O'Reilly.
- Godbolt, Micah (2016): Frontend Architecture for Design Systems: A Modern Blueprint for Scalable and Sustainable Websites. O'Reilly.
- Heimann, Monika und Michale Schütz (2017): Wie Design wirkt: Prinzipien elfolgreicher Gestaltung. Rheinwerk Verlag.
- Krug, Steve (2014): Don't make me think! Web & Mobile Usability: Das intuitive Web. mitp Verlag.
- Nielsen, Jakob (2001): Designing Web Usability. New Riders Publishing
- Philips, Bill et al. (2017): Android Programming: The Big Nerd Ranch Guide. 3. Auflage. Pearson Technology Group.
- Shneiderman, Ben et al. (2017): Designing the User Interface: Strategies for Effective Human-Computer Interaction. 6. Auflage. Pearson.
- Tidwell, Jenifer (2010): Designing Interfaces: Patterns for Effective Interaction Design. 2. Auflage. O'Reilly.
- Vollmer, Guy (2017): Mobile App Engineering. dpunkt Verlag.
  Wolf, Jürgen (2016): HTML5 und CSS3: Das umfassende Handbuch zum Lernen und Nachschlagen. 2. Auflage. Rheinwerk Verlag.

M 6 Backend-Entwicklung

Modul:	Backend-Entwicklung
Kürzel:	M 6
Untertitel:	Wahlpflichtmodul
Modulverantwortlicher:	Prof. Dr. Martin Schmollinger
Dozent(in):	Prof. Dr. Peter Hertkorn Prof. Dr. Christian Kücherer Andre Lindenberg Prof. Dr. Martin Schmollinger
Sprache:	Deutsch
Zuordnung zum Curriculum:	Pflichtmodul 2. Semester
Lehrform / SWS:	Vorlesung mit integrierten Übungen / 4 SWS
Arbeitsaufwand:	Präsenzstudium 60 Stunden Eigenstudium 90 Stunden
Kreditpunkte:	5 ECTS
Voraussetzungen nach StuPro:	Keine
Empfohlene Voraussetzung:	Alle Module aus Semester 1
Studien-/Prüfungsleistungen/ Prüfungsform:	Projektarbeit.

Modulziele:

Unternehmensanwendungen sind in den letzten Jahren feingranularer geworden und werden in Cloud-Plattformen betrieben. Anstelle von schwergewichtigen, monolithischen Anwendungen tritt eine Vielzahl kleinerer Services, die unabhängig voneinander entwickelt und in Produktion gebracht werden können. Diese Aufteilung soll den Verfall der Architektur verhindern und dafür sorgen, dass Unternehmensanwendungen auch langfristig wartbar und erweiterbar bleiben.

Ziel des Modules ist es den Teilnehmerinnen und Teilnehmern Methoden, Konzepte, Muster, Technologien und Werkzeuge zu vermitteln mit denen Backend-Services dieser Architekturen entwickelt werden können. Das vermittelte Wissen wird anhand einer integrierten Projektarbeit eingeübt.

Angestrebte Lernergebnisse

Kenntnisse:

Kenntnisse zu flexiblen Softwarearchitekturen für Unternehmensanwendungen und aktuelle Varianten von Backend-Services. Teilnehmerinnen und Teilnehmer kennen grundlegende Konzepte und Muster von Programmierplattformen und Frameworks für Backend- Services und haben praktische Kenntnisse zu mind. einer konkreten Programmierplattform oder eines konkreten Frameworks. Sie kennen Technologien zur Kommunikation verschiedener Services. Sie kennen verschiedene Schnittstellentechnologien und kennen bei der Entwicklung die Regeln für gutes API-Design. Sie kennen Testarchitekturen und Teststrategien für von Backend-Services, sowie die Grundlagen der ihrer Sicherheit. Darüber hinaus ist bekannt, wie Backend-Services durch eine Continuous-Delivery/Deployment-Pipeline automatisiert bereitgestellt werden können.

Fertigkeiten:

Die Teilnehmerinnen und Teilnehmer sind in der Lage Backend-Services für ein fachliches Szenario zu entwerfen und zu entwickeln. Dabei setzen sie aktuelle Entwicklungswerkzeuge und –methoden, sowie Entwurfsmuster ein und sichern die Services bei Bedarf ab. Sie sind in der Lage für die für den Anwendungsfall geeignete Schnittstellen-Technologie auszuwählen und einzusetzen. Dabei können sie bei der Implementierung eines Services andere Services und Systeme einbinden. Die Qualität der Services wird durch Konzeption und Umsetzung einer geeigneten Teststrategie, sowie durch Einhaltung von Richtlinien beim API-Design und beim Programmieren sichergestellt. Sie sind in der Lage einen vollautomatischen Build- Test- und Deployprozess für die Backend-Services aufzubauen, der den Service in der Cloud (z.B. einer PaaS) bereitstellt.

Kompetenzen:

LE#	Lernergebnis (LE)	Geprüft durch
LE1	Erfahrung mit professioneller Softwareentwicklungs- und Betriebsumgebungen für Backend-Services.	Projektarbeit
LE2	Praktische Anwendung allgemeiner Konzepte und Muster von Programmierplattformen/Frameworks für die Entwicklung von Backend-Services.	Projektarbeit
LE3	Reflexion und Anwendung nicht-funktionaler Aspekte von Backend-Services (z. B. Sicherheit, Qualität des Quellcodes, durchdachtes API-Designs).	Projektarbeit
LE4	Praktische Anwendung allgemeiner Testverfahren zur Sicherstellung funktionaler Aspekte der Services und Aufbau einer CD-Pipeline.	Projektarbeit
LE5	Erstellung eines eigenen Backend-Services mit einem aktuellen Framework unter Einbindung dritter Services und einer geeigneten Schnittstellen-Technologie	Projektarbeit

Inhalt:

- Motivation: Flexible Softwarearchitekturen für Unternehmensanwendungen.
- Professionelle Softwareentwicklungsumgebung für Backend-Services (Z.B. IDEs, Build-Werkzeuge, Versionsverwaltung).
- Grundlegende Konzepte von Programmierplattformen für Backend-Services (am Beispiel der Java Platform / Java EE / Spring Boot).
- Aufbau einer Continuous-Delivery/Deployment-Pipeline in die Cloud (z.B. GitHub -> TravisCI -> Bluemix).
- Kommunikation zwischen Services und Schnittstellentechnologien
- Richtlinien des API-Designs.
- Testkonzepte, -architekturen und -technologien und die Integration in die CD-Pipleline.
- Grundlegende Sicherheitskonzepte und -technologien.

Medienformen:

Das Modul besteht aus einer Vorlesung in seminaristischem Stil mit Tafelanschrieb, Tageslichtprojektion und PC-Projektion und integrierten Übungen zu den Vorlesungsinhalten. Das Material zu den Veranstaltungen gibt es in elektronischer Form: Folienskript zu den Vorlesungen, Übungsblätter mit Aufgaben.

Literatur:

- Gutierrez, Felipe (2016). Pro Spring Boot. APress.
- Newman, Sam (2015). Building Microservices – Designing fine-grained systems. O'Reilly.
- Preißel Rene; Stachmann, Björn (2016). Git. Dpunkt.verlag.
- Schießer, Marcus; Schmollinger (2014), Martin.Workshop Java EE 7. dpunkt.verlag.
- Spichale, Kai (2016). API-Design: Praxishandbuch für Java- und Webservice-Entwickler. dpunkt-verlag.
- Tilkov, Stefan (2015). REST und HTTP. dpunkt.verlag.
- Weil, Dirk (2015); Java EE 7 – Enterprise-Anwendungsentwicklung leicht gemacht. dpunkt.verlag.
- Wolff, Eberhard (2016). Microservices – Grundlagen flexibler Softwarearchitekturen. dpunkt.verlag.
- Wolff, Eberhard (2016). Continuous Delivery: Der pragmatische Einstieg. dpunkt.verlag.
- Sbarski, Peter (2017) Serverless architectures on AWS, Manning.

M 7 Softwarearchitektur

Modul:	Softwarearchitektur
Kürzel:	M 7
Untertitel:	
Lehrveranstaltungen:	Vorlesung
Modulverantwortlicher:	Prof. Dr. Christian Kücherer
Dozent(in):	Prof. Dr. Peter Hertkorn Prof. Dr. Christian Kücherer André Lindenberg Prof. Dr. Jürgen Münch
Sprache:	Deutsch
Zuordnung zum Curriculum:	Professional Software Engineering, Master, Pflichtfach, 2. Semester
Lehrform / SWS:	Vorlesung / 4 SWS
Arbeitsaufwand:	Präsenzstudium 60 Stunden Eigenstudium 90 Stunden
Kreditpunkte:	5 ECTS
Voraussetzungen nach StuPro:	Keine
Empfohlene Voraussetzung:	Methoden und Technologien professioneller Programmierung, Software Engineering
Studien-/Prüfungsleistungen/ Prüfungsform:	Projektarbeit

Modulziele:

Die Veranstaltung Softwarearchitektur hat zum Ziel, die Grundkenntnisse im Bereich Softwarearchitekturen für ein tiefergehendes Verständnis unterschiedlicher Formen von Softwarearchitekturen zu vermitteln. Im Vordergrund steht dabei die Entwicklung der Strukturen von Software, ausgerichtet an der Domäne, in der später das Softwaresystem eingesetzt werden soll. Von großer Bedeutung sind dabei detaillierte Kenntnisse nichtfunktionaler Anforderungen, die sowohl für die Produktqualität entscheidend sind, als auch eine Softwarearchitektur maßgeblich prägen. Wichtige Aspekte moderner Softwaresysteme sind schnelle und leichtgewichtige Entwicklung sowie die Wiederverwendbarkeit von Funktionalität. Aus diesem Grund werden in der Veranstaltung verschiedene Komponententechnologien im Kontext von Softwarearchitektur betrachtet. Insbesondere werden verschiedene Arten von Architekturen behandelt und durch praktische Anwendung durch die Teilnehmerinnen und Teilnehmer vertieft. Da Softwarearchitekturen immateriell und damit schlecht greifbar sind, wenden die Teilnehmerinnen und Teilnehmer Modellierungssprachen und

Visualisierungstechniken zur Modellierung und Veranschaulichung von Softwarearchitekturen an. Außerdem werden die wichtigsten Methoden zur Analyse von Architekturen vermittelt. Durch Diskussion der Kriterien und Merkmale langlebiger Architekturen sowie deren Dokumentation werden moderne Werte und Anforderungen an Softwarearchitekturen vermittelt. Aspekte der Weiterentwicklung werden mit Prinzipien der evolutionären Architekturen betrachtet und analysiert.

Angestrebte Lernergebnisse

Kenntnisse:

- Grundverständnis der Prinzipien von Domain-Driven Design
- Kenntnis unterschiedlicher Komponententechnologien
- Verständnis des Einflusses nichtfunktionaler Anforderungen auf Softwarearchitekturen
- Kenntnisse des Vorgehens zur Architekturfindung
- Grundlegende Architekturmuster, insbesondere Architekturschichten, reaktive und Microservice-Architekturen.
- Verständnis der Vorteile und Anwendung von Modellierungssprachen sowie Architekturvisualisierung mit Software-Cities, Software-Tomographen und anderen bildgebenden Werkzeuge.
- Kenntnis der Methoden zur Analyse von Architekturen
- Kenntnis relevanter Standards zur Dokumentation von Architekturen

Fertigkeiten:

- Die Teilnehmerinnen und Teilnehmer sind in der Lage, selbstständig den Entwurf von komponentenbasierten Softwarearchitekturen nach gegebenen domänenspezifischen Anforderungen und Qualitätsanforderungen zu erstellen.
- Verfeinerung von nichtfunktionalen Anforderungen während des Entwurfs.
- Vergleich und Auswahl unterschiedlicher Architekturmuster
- Anwendung von Modellierungssprachen sowie Werkzeugen zur Visualisierung von Softwarearchitekturen und Interpretation der Ergebnisse.
- Selbständige Durchführung von Architektur-Assessments zur Validierung von Softwarearchitekturen.
- Anwendung von Templates und Standards zur Dokumentation von Architekturen.

Kompetenzen:

Nach Abschluss des Moduls sind die Teilnehmerinnen und Teilnehmer in der Lage

LE#	Lernergebnis (LE)	Geprüft durch
LE1	Domänenspezifische und nichtfunktionale Anforderungen zu analysieren und einzuordnen.	Artefakt
LE2	Komponenten zu entwerfen und deren Schnittstellen zu dokumentieren.	Artefakt

LE3	Softwarearchitekturen ausgehend von Anforderungen und unterschiedlicher Vorgehen zur Architekturfindung selbständig zu entwerfen.	Artefakt
LE4	Alternativen beim Einsatz von Architekturmustern bewerten und anwenden zu können.	Artefakt
LE5	Methoden und Verfahren zur Modellierung und Visualisierung von Softwarearchitekturen anzuwenden und die Ergebnisse zu interpretieren.	Artefakt
LE6	Eine Evaluation einer Softwarearchitektur mittels geeigneter Methoden durchführen zu können.	Artefakt
LE7	Softwarearchitekturen anhand von Templates und Standards zu dokumentieren.	Artefakt

Inhalt:

Zu Beginn der Veranstaltung tauchen die Teilnehmerinnen und Teilnehmer in die Prinzipien des Domain-Driven Design ein und vertiefen Ihre Kenntnisse der nichtfunktionalen Anforderungen im Hinblick auf deren Relevanz für die Qualität eines Softwaresystems (LE1). Durch die Vermittlung der Prinzipien von Komponententechnologien und Komponentenframeworks zur Realisierung von Systembausteinen wenden die Teilnehmerinnen und Teilnehmer diese Prinzipien anhand einer gegebenen Problemstellung an (LE2). Als weiterer Theoriebaustein werden verschiedene Methoden zur Architekturfindung vermittelt und unmittelbar am Beispiel geübt. Ausgehend von den Anforderungen werden nun nach TOGAF, Zachman, dem 4+1 Modell nach Kruchten, sowie dem C4 – Modell für Software-Architekturen eine neue Architektur definiert und die Design-Entscheidungen begründet (LE3). Anschließend werden wichtige Architekturmuster (Patterns) wie, Schichten, nachrichtenbasierte und reaktive Architekturen sowie Microservice-Architekturen betrachtet und diskutiert. Zentrale Konzepte wie Monitoring, Tracing, und Skalierung von Architekturen werden dabei ebenso besprochen, wie Prinzipien der Datenhaltung (LE4). Des Weiteren wird die Architektur eines bestehenden Open-Source Systems mit einem Software-Tomographen bzw. durch Software-Cities visualisiert. Die dabei gewonnenen Erkenntnisse werden von den Teilnehmerinnen und Teilnehmern präsentiert und diskutiert (LE5).

Zur Evaluation von Software-Architekturen werden die Prinzipien des Architektur-Assessment nach ATAM oder SAAM vermittelt. Teilnehmerinnen und Teilnehmer lernen an einem realen Beispiel eine Architekturevaluation durchzuführen (LE6). Anschließend werden die Anforderungen an Software-Architekturen hinsichtlich ihrer Weiterentwicklung, Stabilität, Robustheit und Erweiterbarkeit im Rahmen evolutionärer Software-Architekturen besprochen. Die Dokumentation von Architekturen wird mit verschiedenen Templates und Standards, wie z.B. dem arc42 Template oder der ISO/IEC/IEEE 42010 Standard besprochen und exemplarisch von den Teilnehmerinnen und Teilnehmern geübt (LE7).

Medienformen:

Das Lehrmaterial besteht aus einem Folienskript, das in elektronischer Form vorliegt, Übungsblättern sowie Programmbeispielen. Seminaristischer Unterricht mit Whiteboard, PC-Beamer und Präsentationsfolien, bei dem Beispiele zu den theoretischen Inhalten veranschaulicht werden sowie Demonstration von Beispielprogrammen und interaktiver Programmentwicklung. Die Teilnehmerinnen und Teilnehmer bearbeiten individuell oder in Gruppen Übungsaufgaben zum Themengebiet Softwarearchitektur mit Betreuung durch DozentInnen.

Literatur:

- Bass, Len et al. (2013): Software architecture in practice. 3. Auflage. Upper Saddle River, NJ ; Munich [u.a.] : Addison-Wesley
- Clements, Paul et al. (2011): Evaluating Software Architectures: Methods and Case Studies. Boston, Mass.; Munich [u.a.] Addison-Wesley.
- Eilebrecht, Karl und Starke Gernot (2019): Patterns kompakt: Entwurfsmuster für effektive Software-Entwicklung. 5. Auflage. Berlin, Heidelberg: Springer Vieweg.
- Evans, Eric (2009): Domain Driven Design - Tackling Complexity in the Heart of Business Software. Boston; Munich [u.a.]: Addison-Wesley.
- Fowler, Martin (2002): Patterns of Enterprise Application Architecture. Addison-Wesley.
- Gharbi, Mahbouba et al. (2018): Basiswissen für Softwarearchitekten: Aus- und Weiterbildung nach iSAQB-Standard zum Certified Professional for Software Architecture - Foundation Level. 3. Auflage. Heidelberg, dpunkt.
- Goll, Joachim (2018): Entwurfsprinzipien und Konstruktionskonzepte der Softwaretechnik: Strategien für schwach gekoppelte, korrekte und stabile Software. Wiesbaden: Springer Vieweg.
- Reussner, Ralf (Hg.) (2009): Handbuch der Software-Architektur. 2. Auflage. Heidelberg: dpunkt.
- Lilienthal, Carola (2017): Langlebige Softwarearchitekturen: technische Schulden analysieren, begrenzen und abbauen. 2. Auflage. Heidelberg: dpunkt.
- Starke, Gernot (2018): Effektive Softwarearchitekturen: ein praktischer Leitfaden. 8. Auflage. München: Hanser.
- Vogel, Oliver et a. (2009): Software-Architektur: Grundlagen – Konzepte – Praxis. 2. Auflage. Spektrum Akademischer Verlag.

M 8 Softwareprojekt 1

Modul:	Softwareprojekt 1
Kürzel:	M 8
Untertitel:	
Lehrveranstaltungen:	Projekt
Modulverantwortlicher:	Prof. Dr. Martin Schmollinger
Dozent(in):	Betreuer aus Unternehmen, ggf. auch Professorinnen und Professoren des Programms
Sprache:	Deutsch
Zuordnung zum Curriculum:	Pflichtfach, 2. Semester
Lehrform / SWS:	Projekt / Der Anteil der Präsenzzeit variiert stark in Abhängigkeit von der gewählten Projektmethode und -organisation.
Arbeitsaufwand:	150 Stunden
Kreditpunkte:	5 ECTS
Voraussetzungen nach StuPro:	Keine
Empfohlene Voraussetzung:	Alle Module des 1. Semesters.
Studien-/Prüfungsleistungen/ Prüfungsform:	Projektarbeit, Referat

Modulziele:

Im Rahmen des Masterprogramms wird ein Softwareprojekt durchgeführt. In diesem kann das erlernte Wissen anwendungsnah vertieft werden. Didaktisch wird im Modul Softwareprojekt 1 die Grundlage für eine erfolgreiche Durchführung des Projekts im Modul Softwareprojekt 2 gelegt.

Im Modul Softwareprojekt 1 sollen die Teilnehmerinnen und Teilnehmer ihre bisher erworbenen fachlichen Kompetenzen zum Aufsetzen eines umfassenden Softwareprojekts gesamthaft zur Anwendung bringen und praxisorientiert weiterentwickeln. Des Weiteren sollen Sie den Projektauftrag im Hinblick auf ethische Fragestellungen reflektieren und mit den Dozenten diskutieren.

Außerdem sollen bereits in der Aufbauphase des Projekts Überlegungen angestellt werden wie Datenschutz in der zu erstellenden Software und während der Entwicklung gewährleistet werden kann.

Angestrebte Lernergebnisse

Kenntnisse:

Die Teilnehmerinnen und Teilnehmer kennen die wissenschaftlichen und praxisorientierten Methoden zur Organisation und zum Management von Softwareprojekten. Sie kennen unterstützende Werkzeuge des Softwareprojekt-Managements. Sie sind mit grundlegenden Themen aus der jeweiligen Anwendungsdomäne vertraut. Sie kennen Systeme, Softwarearchitekturen, Technologien und Frameworks zur Entwicklung der Software. Sie kennen die Grundzüge der Berufsethik in der Softwareentwicklung und die Konzepte Verantwortung, Wert, Dilemma.

Fertigkeiten:

Die Teilnehmerinnen und Teilnehmer können anhand einschlägiger Methoden eine Projektorganisation systematisch erstellen und dabei eine Arbeitsteilung und Arbeitszusammenführung berücksichtigen. Sie können den Arbeitsstand angemessen dokumentieren und kommunizieren und ggf. den Arbeits- und Zeitplan anpassen. Sie können die für die Durchführung des Projektes notwendigen Technologien, Frameworks und Werkzeuge verwenden. Sie sind in der Lage die verwendeten Muster und Softwarearchitekturen zu erklären. Sie können Maßnahmen zum Schutz personenbezogener Daten erklären. Sie können die Ergebnisse der Projektgruppe vor einem fachkundigen Publikum und mit fachspezifischer visueller Unterstützung auf hohem Niveau präsentieren.

Kompetenzen:

Die Teilnehmerinnen und Teilnehmer verfügen über solide fachliche Kompetenzen und verbesserte Methodenkompetenzen. Die Kommunikations- und Teamfähigkeit der Teilnehmerinnen und Teilnehmern ist weiterentwickelt. Ihre Sozial- und Problemlösungskompetenz ist gestärkt. Sie können Herausforderungen der Berufsethik in der Arbeit von Softwareentwicklerinnen und -entwicklern in einer konkreten Projektsituation erkennen.

Inhalt:

Durch den Dozenten/Betreuer moderiertes aber weitestgehend eigenständiges Aufsetzen eines Softwareprojekts für eine umfassende Aufgabenstellung aus einer beliebigen Anwendungsdomäne. Im Rahmen des Moduls werden Methoden, Prozesse, Rollen, Infrastrukturen, Werkzeuge, Frameworks, sowie Technologien und Systeme vorgestellt, ausgewählt, aufgebaut oder vertieft. Darüber hinaus werden zur Übung auch schon erste einfache Inkremente des Softwareprodukts erstellt. Das Projekt wird in Gruppen von ca. 4 – 6 Teilnehmerinnen und Teilnehmern. Das erarbeitete Projektergebnis und die Projekterfahrungen sind in Form einer Abschlusspräsentation einem fachkundigen Publikum vorzustellen und in einer anschließenden Diskussion zu verteidigen. Die Projektergebnisse sind in einer Projektdokumentation gesamthaft zu erfassen.

Medienformen: Projektspezifische Medienformen.

Literatur: Abhängig von der jeweiligen Aufgabenstellung

M 9 Softwareprojekt 2

Modul:	Softwareprojekt 2
Kürzel:	M 9
Untertitel:	
Lehrveranstaltungen:	Projekt
Modulverantwortlicher:	Prof. Dr. Martin Schmollinger
Dozent(in):	Betreuer aus Unternehmen, ggf. auch Professorinnen und Professoren des Programms
Sprache:	Deutsch
Zuordnung zum Curriculum:	Pflichtfach 3. Semester
Lehrform / SWS:	Projekt / Der Anteil der Präsenzzeit variiert stark in Abhängigkeit von der gewählten Projektmethode und -organisation.
Arbeitsaufwand:	300 Stunden insgesamt
Kreditpunkte:	10 ECTS
Voraussetzungen nach StuPro:	Keine
Empfohlene Voraussetzung:	Softwareprojekt 1
Studien-/Prüfungsleistungen/ Prüfungsform:	Projektarbeit, Referat

Modulziele:

Im Rahmen des Masterprogramms wird ein Softwareprojekt durchgeführt. In diesem kann das erlernte Wissen anwendungsnah vertieft werden. Didaktisch wird im Modul Softwareprojekt 1 die Grundlage für eine erfolgreiche Durchführung des Projekts im Modul Softwareprojekt 2 gelegt.

In Modul Softwareprojekt 2 sollen die Teilnehmerinnen und Teilnehmer aufbauend auf ihren bisher erworbenen methodischen und fachlichen Kompetenzen ein Softwareprojekt durchführen. Ziel ist es am Ende des Moduls ein Softwareprodukt mit modernen Methoden und Technologien des Software-Engineering in einem Entwicklerteam erstellt zu haben.

Angestrebte Lernergebnisse

Kenntnisse:

Die Teilnehmerinnen und Teilnehmer kennen die auftretende Komplexität bei der Anwendung wissenschaftlicher und praxisorientierten Methoden zur Durchführung von Softwareprojekten. Sie kennen Details bei der Verwendung unterstützender Werkzeugen des Softwareprojekt-Managements. Sie sind mit fortgeschrittenen Themen aus der jeweiligen Anwendungsdomäne vertraut. Sie kennen Details der verwendeten Programmierframeworks, Systeme, Softwarearchitektur und Muster.

Fertigkeiten:

Die Teilnehmerinnen und Teilnehmer sind mit den Aufgaben der verschiedenen Rollen im Rahmen von Softwareentwicklungsmethoden vertraut und können diese selbst einnehmen. Sie sind mit Verfahren zur Aufwandsabschätzungen und Priorisierung von Aufgaben vertraut und haben Erfahrungen bei ihrer Anwendung gesammelt. Sie können die für die Durchführung des Projektes notwendigen Technologien, Frameworks und Werkzeuge sicher verwenden. Sie sind in der Lage die verwendeten Muster und Softwarearchitekturen zu erklären und anzuwenden. Sie können die Ergebnisse der Projektgruppe vor einem fachkundigen Publikum und mit fachspezifischer visueller Unterstützung auf hohem Niveau präsentieren.

Kompetenzen:

Die Teilnehmerinnen und Teilnehmer verfügen über fortgeschrittene fachliche Kompetenzen und erweiterte Methodenkompetenz. Die Kommunikations- und Teamfähigkeit der Teilnehmerinnen und Teilnehmern ist wesentlich weiterentwickelt. Ihre Sozial- und Problemlösungskompetenz ist gestärkt.

Inhalt:

Im Modul Softwareprojekt 2 liegt der Schwerpunkt auf der Durchführung eines Softwareprojekts. Dabei werden auch weiterführende Themen wie z. B. der Skalierung der Software betrachtet.

Durch den Dozenten/Betreuer teilweise moderiertes aber weitgehend eigenständiges Softwareprojekt für eine umfassende Aufgabenstellung aus einer beliebigen Anwendungsdomäne. Die Bearbeitung der Aufgabenstellung erfolgt in einem Team von Teilnehmerinnen und Teilnehmern. Die Regelgröße des Teams beträgt 4-6 Entwickler. Das erarbeitete Projektergebnis und die Projekterfahrungen sind in Form einer Abschlusspräsentation einem fachkundigen Publikum vorzustellen und in einer anschließenden Diskussion zu verteidigen. Die Projektergebnisse sind in einer Projektdokumentation gesamthaft zu erfassen.

Medienformen:

Projektspezifische Medienformen.

Literatur:

Abhängig von der jeweiligen Aufgabenstellung

M 10/M 11 Wahlpflichtmodul 1 und 2

W1 Distributed Ledger Technology

Modul:	Distributed Ledger Technology
Kürzel:	M 10 / M11 W1
Untertitel:	
Lehrveranstaltungen:	Vorlesung
Modulverantwortlicher:	Prof. Dr. Marcus Schöller
Dozent(in):	Prof. Dr. Peter Hertkorn Prof. Dr. Marcus Schöller
Sprache:	Deutsch
Zuordnung zum Curriculum:	Professional Software Engineering Master, Wahlfach, 3. Semester
Lehrform / SWS:	Vorlesung 4 SWS
Arbeitsaufwand:	Präsenzstudium 60 Stunden Eigenstudium 90 Stunden
Kreditpunkte:	5 ECTS
Voraussetzungen nach StuPro:	Keine
Empfohlene Voraussetzung:	Keine
Studien-/Prüfungsleistungen/ Prüfungsform:	Projektarbeit

Modulziele:

Blockchain-Technologien sind die aktuelle Ausprägung zur Lösung verteilter, konsistenter und vertrauenswürdiger Datenhaltung. Sie gehören zu einer breiteren Gruppe von Lösungen: Distributed Ledger Technology (DLT). In diesem Modul werden die zugrundeliegenden Konzepte sowie ausgewählte Methoden und Verfahren im Bereich DLT behandelt, aktuelle Entwicklungen in diesem Bereich besprochen sowie deren Einsatzmöglichkeiten untersucht und bewertet. Mit einem erfolgreichen Bestehen des Moduls soll sichergestellt sein, dass die Studierenden in der Lage sind, den Einsatz von DLT abzuwägen, geeignete Technologien auswählen, verstehen und zur Entwicklung von Anwendungen verwenden zu können. Darauf aufbauend sollen die Studierenden Kenntnisse über den Betrieb von DLT-Anwendungen erwerben.

Angestrebte Lernergebnisse:

Kenntnisse:
Absolventen dieses Moduls verfügen nach dem erfolgreichen Abschluss über Kenntnisse der zugrunde liegenden Konzepte von DLT-Lösungen und können die Unterschiede zwischen zentralisierten und dezentralisierten Systemen erläutern. Sie haben ein Verständnis entwickelt für technische und anwendungsspezifische Aspekte von DLT, insbesondere unterschiedliche Konsensusprotokolle, die Ebenen dezentralisierter Anwendungen und verschiedene Technologien für Distributed Ledger. Weiter haben sie Kenntnisse über DLT-basierte Anwendungen und deren Entwicklung uns sind in der Lage, verschiedene Verfahren zur Erstellung dezentralisierter Anwendungen zu beschreiben, zu bewerten und zu nutzen. Die Studierenden kennen aktuelle Trends und Entwicklungen im Bereich DLT und können deren Auswirkung auf das Gebiet DLT einschätzen und einordnen.

Fertigkeiten:
Die Studierenden können gegebene Problemstellungen im Hinblick auf den Einsatz von DLT analysieren, geeignete Technologien auswählen und Anwendung auf Basis von DLT entwerfen und entwickeln. Dafür kennen Sie existierende DLT-Implementierungen und die Anforderungen an den Betrieb. Die Studierenden können Lösungen ganzheitlich analysieren und bewerten und somit technisch fundierte Entscheidungen für die Einsatzmöglichkeiten dieser Technologien treffen.

Kompetenzen:
Nach Abschluss des Moduls sind die Studierenden in der Lage:

LE#	Lernergebnis (LE)	Geprüft durch
LE1	DLT-Grundlagen kennen und beurteilen zu können.	Artefakt
LE2	Komponenten und deren Aufgaben in DLT-Lösungen in Beziehung setzen zu können.	Artefakt
LE3	Die Anwendung von DLT verstehen und bewerten zu können.	Artefakt
LE4	Methoden der Softwareentwicklung für DLT verstehen und anwenden zu können.	Artefakt

Inhalt:

Zuerst werden grundlegende Konzepte von DLT eingeführt: Konsensprotokoll, Kryptographie und Smart Contracts (LE1). Darauf aufbauend werden verschiedene DLT-Lösungen untersucht und verglichen, z.B. Bitcoin, Ethereum, Hyperledger (LE2, LE3). Zuletzt sollen konkrete Anwendungsszenarien für DLT entwickelt und prototypisch umgesetzt werden (LE4).

Medienformen:

Das Lehrmaterial besteht aus einem Folienskript, das in elektronischer Form vorliegt, Übungsblättern sowie Programmbeispielen. Seminaristischer Unterricht mit Whiteboard, PC-Beamer und Präsentationsfolien, bei dem Beispiele zu den theoretischen Inhalten veranschaulicht werden sowie Demonstration von Beispielprogrammen und interaktiver Programmentwicklung. Die Studierenden bearbeiten individuell oder in Gruppen Übungsaufgaben zum Themengebiet DLT mit Betreuung durch den Dozenten.

Literatur:

- Antonopoulos, Andreas M. (2017): Mastering Bitcoin. 2. Auflage. O'Reilly.
- Antonopoulos, Andreas M. und Gavin Wood (2018): Mastering Ethereum: Building Smart Contracts and Dapps. O'Reilly.
- Cachin, Christian und Marko Vukolic (2017): "Blockchain Consensus Protocols in the Wild", CoRR, eprint arXiv:1707.01873, https://dblp.org/rec/bib/journals/corr/CachinV17.
- Drescher, Daniel (2017): Blockchain Basics: A Non-Technical Introduction in 25 Steps. Apress.
- Kosba, Ahmed et al. (2016): "Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts," 2016 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 2016, S. 839-858.
- Laurence, Tiana (2017): Blockchain for Dummies. For Dummies.
- Raval, Siraj (2016): Decentralized Applications: Harnessing Bitcoin's Blockchain Technology. O'Reilly.
- Swan, Melanie (2015): Blockchain: Blueprint for a New Economy. O'Reilly.
- Tapscott, Don und Alex Tapscott (2016): Blockchain Revolution: How the Technology Behind Bitcoin Is Changing Money, Business, and the World. Portfolio.
- Wattenhofer, Roger (2017): Distributed Ledger Technology: The Science of the Blockchain. 2. Auflage. CreateSpace Independent Publishing.

W2 Big Data-Technologien

Modul:	Big Data-Technologien
Kürzel:	M10/M11 W2
Untertitel:	
Lehrveranstaltungen:	Vorlesung
Studiensemester:	jedes Semester
Modulverantwortlicher:	Prof. Dr. Peter Hertkorn
Dozent(in):	Prof. Dr. Peter Hertkorn Prof. Dr. Wolfgang Blochinger
Sprache:	Deutsch
Zuordnung zum Curriculum:	Professional Software Engineering Master, Wahlfach, 3. Semester
Lehrform / SWS:	Vorlesung, Blockveranstaltung / 4 SWS
Arbeitsaufwand:	Präsenzstudium 60 Stunden Eigenstudium 90 Stunden
Kreditpunkte:	5 ECTS
Voraussetzungen nach StuPro:	Keine
Empfohlene Voraussetzung:	Datenbanksysteme
Studien-/Prüfungsleistungen/ Prüfungsform:	Projektarbeit

Modulziele:

Das Ziel des Moduls ist es, die Teilnehmerinnen und Teilnehmer für die Technologien im Themengebiet Big Data zu sensibilisieren und sie in die Lage zu versetzen, Big Data-Architekturen zu entwickeln und die dabei eingesetzten Technologien zu verstehen. In diesem Modul werden ausgewählte Technologien im Bereich Big Data behandelt, wobei die Kompetenzen aus anderen Modulen eingesetzt und vertieft werden können. Es werden aktuelle Entwicklungen im Bereich Big Data-Technologien besprochen und deren Einsatzmöglichkeiten untersucht und bewertet. Mit einem erfolgreichen Bestehen des Moduls soll sichergestellt sein, dass die Teilnehmerinnen und Teilnehmer in der Lage sind, Big Data-Architekturen zu entwickeln sowie geeignete Technologien auswählen, verstehen und bedienen zu können.

Angestrebte Lernergebnisse

Kenntnisse:

Die Teilnehmerinnen und Teilnehmer

- sind in der Lage, die Elemente einer Big Data-Architektur zu erläutern.
- kennen die Prinzipien, um robuste und skalierbare Datensysteme zu entwerfen.
- kennen die unterschiedlichen Ebenen einer Big Data-Architektur.
- können unterschiedliche Methoden zur Speicherung von Daten in Big Data-Architekturen erklären, bewerten und nutzen.
- kennen das Programmiermodell MapReduce.
- kennen unterschiedliche Methoden der Batchverarbeitung.
- sind in der Lage, verschiedene Verfahren zur Echtzeitverarbeitung von Daten zu beschreiben, zu bewerten und zu nutzen.
- kennen die Bedeutung und Methoden der Datenanalyse.
- kennen aktuelle Trends und Entwicklungen im Bereich Big Data-Technologien und können deren Auswirkung auf das Gebiet Big Data einschätzen und einordnen.

Fertigkeiten:

Die Teilnehmerinnen und Teilnehmer bauen eine Big Data-Architektur auf Basis von Hadoop auf. Sie erstellen Programme zur Analyse von Daten mittels MapReduce und führen diese auf einem Hadoop-Cluster aus. Die Teilnehmerinnen und Teilnehmer analysieren die Anforderungen für gegebene Problemstellungen, wählen Technologien für die einzelnen Ebenen einer Lambda-Architektur aus und wenden diese im Rahmen gegebener Szenarien an. Sie vergleichen die Eigenschaften einer Lambda-Architektur mit denen einer Kappa-Architektur und bauen eine Streaming-Architektur mittels ausgewählter Technologien auf. Auf Basis der jeweiligen Architekturen wenden die Teilnehmerinnen und Teilnehmer verschiedene Verfahren zur Analyse von Daten an.

Kompetenzen:

Nach Abschluss des Moduls sind die Teilnehmerinnen und Teilnehmer in der Lage:

LE#	Lernergebnis (LE)	Geprüft durch
LE1	Big Data-Architekturen planen und erstellen zu können.	Artefakt
LE2	Technologien zur Batchverarbeitung verstehen und produktiv einzusetzen.	Artefakt
LE3	Technologien zur Echtzeiterarbeitung von Daten verstehen und produktiv einzusetzen.	Artefakt
LE4	Verfahren zur Analyse von Daten analysieren und deren Vor- und Nachteile zu bewerten.	Artefakt
LE5	Unterschiedliche Technologien für ein gegebenes Anwendungsszenario bewerten und geeignete Technologien auswählen zu können.	Artefakt
LE6	Probleme und Grenzen von Technologien im Bereich Big Data einschätzen zu können.	Artefakt
LE7	Aktuelle Entwicklungen im Bereich Big Data beurteilen und sich anzueignen.	Artefakt

Inhalt:

In der Vorlesung werden die Teilnehmerinnen und Teilnehmer über den Einsatz von Big Data-Technologien und grundlegende Architekturmodelle an das Themengebiet herangeführt (LE1). Entsprechend der verschiedenen Ebenen einer Lambda-Architektur werden unterschiedliche Technologien ausführlich behandelt, deren Eigenschaften analysiert, bewertet und auf gegebene Problemstellungen angewendet (LE2-6). Schwerpunkte bilden dabei die Technologien zur Batchverarbeitung, zur Echtzeitverarbeitung von Daten sowie Methoden und Verfahren der Datenanalyse. Darauf aufbauend werden die Elemente einer Streaming-Architektur behandelt, wobei die verschiedenen Technologien miteinander verglichen und im Rahmen von unterschiedlichen Szenarien beleuchtet und angewendet werden (LE5-6). Des Weiteren werden neuere Entwicklungen im Bereich Big Data-Technologien vorgestellt und deren Eigenschaften mit denen der behandelten Technologien verglichen (LE7). Bei der praktischen Umsetzung wird darauf geachtet, dass in der Industrie genutzte Werkzeuge eingesetzt werden, so dass auch ein praktisches Wissen erworben wird.

Medienformen:

Das Lehrmaterial besteht aus einem Folienskript, das in elektronischer Form vorliegt, Übungsblättern sowie Programmbeispielen. Seminaristischer Unterricht mit Whiteboard, PC-Beamer und Präsentationsfolien, bei dem Beispiele zu den theoretischen Inhalten veranschaulicht werden sowie Demonstration von Beispielprogrammen und interaktiver Programmentwicklung. Die Teilnehmerinnen und Teilnehmer bearbeiten individuell oder in Gruppen Übungsaufgaben zum Themengebiet Big Data mit Betreuung durch den Dozenten.

Literatur:

- Akidau, Tyler et al. (2018): Streaming Systems: The What, Where, When, and How of Large-Scale Data Processing. O'Reilly.
- Chambers, Bill und Matei Zaharu (2018): Spark: The Definitive Guide: Big Data Processing Made Simple. O'Reilly
- Freiknecht, Jonas und Stefan Papp (2018): Big Data in der Praxis: Lösungen mit Hadoop, Spark, HBase und Hive. Daten speichern, aufbereiten, visualisieren. 2. Auflage. Hanser.
- George, Lars (2011): HBase: The Definitive Guide. O'Reilly.
- Grover, Mark et al. (2015): Hadoop Application Architectures: Designing Real-World Big Data Applications. O'Reilly.
- Karau, Holden und Andy Konwinski (2015): Learning Spark: Lightning-Fast Big Data Analysis. O'Reilly.
- Kleppmann, Martin (2017): Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. O'Reilly.
- Kunigk, Jan et al. (2018): Hadoop in the Enterprise: Architecture: A Guide to Successful Integration. O'Reilly.
- Marz, Nathan and James Warren (2015): Big Data: Principles and Best Practices of Scalable Realtime Data Systems. Manning.
- Narkhede, Neha et al. (2017): Kafka: The Definitive Guide: Real-time Data and Stream Processing. O'Reilly.
- White, Tom (2015): Hadoop: The Definitive Guide: Storage and Analysis at Internet Scale. 4. Auflage. O'Reilly.

W3 Internet of Things

Modul:	Internet of Things
Kürzel:	M 10/M11 W3
Untertitel:	Internet of Things (IoT)
Lehrveranstaltungen:	Vorlesung mit integrierter Übung
Studiensemester:	3. Semester
Modulverantwortlicher:	Prof. Dr. Christian Decker
Dozent(in):	Prof. Dr. Christian Decker
Sprache:	Deutsch
Zuordnung zum Curriculum:	Wahlpflichtmodul,3. Semester
Lehrform / SWS:	Vorlesung / 4 SWS
Arbeitsaufwand:	Präsenzstudium 60 Stunden Eigenstudium 90 Stunden
Kreditpunkte:	5 ECTS
Voraussetzungen nach StuPro:	Keine
Empfohlene Voraussetzung:	Gute Informatikkenntnisse, insb. Softwareengineering; Kenntnisse im wissenschaftlichen Arbeiten
Studien-/Prüfungsleistungen/ Prüfungsform:	Projektarbeit

Modulziele:

Das Internet der Dinge, engl. Internet of Things (IoT), beschäftigt sich mit der Informationsverarbeitung in Umgebungen, in denen extrem viele miniaturisierte Rechnersysteme miteinander vernetzt sind und mit Benutzern auf vielfältige Weise interagieren können.

Ziel des Moduls ist es, die Teilnehmerinnen und Teilnehmer in die Grundlagen, Technologien und Anwendungsmöglichkeiten des Internet of Things (IoT) einzuführen. Das umfasst ein schichtenübergreifendes Know-How über den Aufbau, Funktionsweise und Vernetzung von Rechnersystemen und deren verteilte Informationsverarbeitung. Dies wird durch die Vermittlung von Wissen in den Bereichen Hardware, Software, Kommunikationsprotokolle, Middleware und Systemdesign erreicht.

Angestrebte Lernergebnisse

Kenntnisse:

- Veränderte Ausprägung der Informationsverarbeitung durch miniaturisierte vernetzte Rechnersysteme
- Wissen über die Technologieanforderungen an Rechnersysteme, die in die reale Welt quasi unsichtbar eingebettet sind
- Kommunikationstechnologien und –protokolle zur massiven Vernetzung von eingebetteten Rechnersystemen
- Möglichkeiten und Einsatz von Sensorik
- Klassifikation von IoT Anwendungen und Entwicklungsmethoden
- IoT Systemdesign, Plattformen und Kommunikationsmustern integrierender Systeme
- Value Driver und Veränderungen von Geschäftsmodellen durch IoT
- Web als Middleware im Web-of-Things (WoT)

Fertigkeiten:

Die Teilnehmerinnen und Teilnehmer werden in die Lage versetzt, selbständig auf verschiedenen Ebenen im Unternehmen IoT Anwendungen zu entwerfen und zu entwickeln. Sie entwickeln ein schichtenübergreifendes Verständnis von Rechnersystemen und deren vernetzte Informationsverarbeitung im Zusammenspiel mit neuen Möglichkeiten der impliziten Benutzerinteraktion. Dazu gehört die Fertigkeit zugehörige Managementfunktionen ausüben und IoT Ansätze erfolgreich in Unternehmensanwendungen zu integrieren.

Kompetenzen:

Lernergebnis (LE)	Geprüft durch
Die Teilnehmerinnen und Teilnehmer sind in der Lage, IoT Lösungen zu entwerfen, grundlegende Konzepte der Einbettung durch Smart Object Computer, Designprinzipien von Anwendungen in der Lösung explizit darzustellen und korrekt einzuordnen. Durch ein schichtenübergreifendes Verständnis von Rechnersystemen haben sie die Kompetenz die Schlüsseleigenschaften von IoT Technologien einzuschätzen, um neuartige oder verbesserte Anwendungen durch die massive Vernetzung von eingebetteter Informationstechnologie und deren Services zu verwirklichen. Schließlich können Die Teilnehmerinnen und Teilnehmer die Lösungen hinsichtlich ihres geschäftsrelevanten Beitrages bewerten.	Projektarbeit

Inhalt:

Das Modul vermittelt die Grundlagen und Konzepte des Themenfeldes „Internet der Dinge“. Es werden Hardware- und Softwaretechnologien, insbesondere zur sensorischen Erfassung und Kommunikationsprotokolle, besprochen. Schwerpunkte bilden die Themenbereiche Smart Object Computer, IoT Plattformen, Anwendungen und Entwicklungsmethoden sowie das Web of Things. Kleinere Aufgaben während der Vorlesung vertiefen die Inhalte. Das Modul behandelt folgende Themenbereiche:

- Einführung und Einordnung in die Entwicklung der Computertechnologie
- Enabling Technologie, Einbettung „The invisible computer“, Smart Object Computer
- Kommunikationsformen von IoT Technologien und sensorische Erfassung
- IoT Anwendungen und Entwicklungsmethoden
- IoT Geschäftsmodelle
- IoT Plattformen für die Integration mit weiteren informationsverarbeitenden Systemen
- Web of Things (WoT)

Medienformen:

PDF der Folien aus der Vorlesung. Weiteres Material wird während der Vorlesung bekannt gegeben.

Literatur:

- Weiser, M. The computer for the 21st century
- Mattern F., Flörkemeier, Ch. Vom Internet der Computer zum Internet der Dinge. Informatik Spektrum, Vol. 33, no. 2, S. 107-121, April 2010
- Porter, M.E., Heppelmann, J.E., How Smart, Connected Products Are Transforming Competition. Harvard Business Review 92, no. 11, S. 64-88, November 2014

M 12 Master Thesis

Modul:	Master-Thesis
Kürzel:	M 12
Untertitel:	
Lehrveranstaltungen:	Master Thesis
Studiensemester:	Jedes
Modulverantwortlicher:	Prof. Dr. Martin Schmollinger
Dozent(in):	Alle am Masterprogramm beteiligten Professorinnen und Professoren
Sprache:	Deutsch
Zuordnung zum Curriculum:	Pflichtfach, 4. Semester
Lehrform / SWS:	Master-Thesis / Keine verpflichtende Präsenzzeit
Arbeitsaufwand:	900 Stunden
Kreditpunkte:	30 ECTS
Voraussetzungen nach StuPro:	50 ECTS bei Vorstudium mit mind. 210 ECTS, sonst 80 ECTS
Empfohlene Voraussetzung:	Alle anderen Lehrveranstaltungen des Masterprogramms Professional Software Engineering
Studien-/Prüfungsleistungen/ Prüfungsform:	Master-Thesis / Kolloquium

Modulziele:

Die Master-Thesis ist eine abschließende Prüfungsarbeit, mit der die Teilnehmerinnen und Teilnehmern nachweisen, dass sie eine umfassende Aufgabenstellung der Informatik selbstständig nach grundlegenden wissenschaftlichen Methoden in einem vorgegebenen Zeitrahmen bearbeiten und das Vorgehen und die erreichten Ziele verteidigen können.

Angestrebte Lernergebnisse

Kenntnisse:

Die Teilnehmerinnen und Teilnehmer verfügen über umfassende Kenntnisse aus dem Themenbereich der Arbeit. Die Teilnehmerinnen und Teilnehmern sind mit allen formalen Anforderungen für das Erstellen von wissenschaftlichen Arbeiten vertraut.

Fertigkeiten:

Die Teilnehmerinnen und Teilnehmer können ein abgeschlossenes Gebiet eigenständig nach wissenschaftlichen Methoden bearbeiten. Sie beherrschen einschlägige Techniken für das Anfertigen einer wissenschaftlichen Arbeit wie Gliederung, Zitieren und Einhaltung einer adäquaten äußeren Form.

Kompetenzen:

Die Teilnehmerinnen und Teilnehmer sind zur Abstraktion und Modellbildung zum Zweck der praktischen Analyse, Konzeption und Gestaltung befähigt. Sie verfügen über Analyse-, Design-, Realisierungs- und Projektmanagementkompetenz. Sie sind zur zielorientierten Lösungsentwicklung in der Lage.

Inhalt:

Themen von Master-Arbeiten beziehen sich auf Aufgabenstellungen der Informatik, die aktuell und für die absehbare Zukunft in der Disziplin relevant sind. Die Themen beinhalten mehrere informatische, softwaretechnische, mediale, psychologische, didaktische, wirtschaftliche oder andere Aspekte, die in einem komplexen Zusammenhang mit der Lösung der Aufgabe stehen.

Medienformen:

Fachliche und methodische Betreuung der Teilnehmerinnen und Teilnehmer durch Beratungs- und Betreuungsgespräche, die bei unternehmensnahen Arbeiten auch vor Ort stattfinden. Für die Teilnehmerinnen und Teilnehmer ergibt sich darüber hinaus die Notwendigkeit, relevante Informationen zu recherchieren und zu referenzieren sowie ggf. die Relevanz und Zielorientierung im betrieblichen Umfeld nachzuweisen. Präsentationen der Teilnehmerinnen und Teilnehmer hinsichtlich des Arbeitsfortschrittes und der weiteren Planung.

Literatur:

Abhängig von der jeweiligen Aufgabenstellung