IHK Sachliche Gliederung de Berufsausbildung

Relevante Lernfelder der Berufsschule

- LF 5: Software zur Verwaltung von Daten anpassen
- LF 8: Daten systemübergreifend bereitstellen
- LF 10a: Benutzerschnittstellen gestalten und entwickeln
- LF 11a: Funktionalität in Anwendungen realisieren
- LF 12a: Kundenspezifische Anwendungsentwicklung durchführen

Konkrete Prüfungsthemen

Informieren und Beraten von Kunden und Kundinnen

- Gespräche situationsgerecht führen und Kunden und Kundinnen unter Berücksichtigung der Kundeninteressen beraten
- Kundenbeziehungen unter Beachtung rechtlicher Regelungen und betrieblicher Grundsätze gestalten
- Daten und Sachverhalte interpretieren, multimedial aufbereiten und situationsgerecht unter Nutzung digitaler Werkzeuge und unter Berücksichtigung der betrieblichen Vorgaben präsentieren
- Aktives Zuhören, Kommunikationsmodelle (z.B. Telefonkonferenzen, Chat, virtuelle Teambesprechung), Verkaufsgespräche (Anfrage, Angebot, Auftrag)
- Kundenbeziehungen unter Beachtung rechtlicher Regelungen und betrieblicher Grundsätze gestalten
- Instrumente zur Datenauswertung kennen und bedarfsgerecht auswählen sowie Ergebnisse interpretieren können

Beurteilen marktgängiger IT-Systeme und kundenspezifischer Lösungen

- technologische Entwicklungstrends von IT-Systemen feststellen sowie ihre wirtschaftlichen, sozialen und beruflichen Auswirkungen aufzeigen
- Veränderungen von Einsatzfeldern für IT-Systeme aufgrund technischer, wirtschaftlicher und gesellschaftlicher Entwicklungen feststellen
- Chancen und Risiken der technischen Entwicklungen kennen und identifizieren können
- Veränderungen von Einsatzfeldern kennen und beurteilen können

Entwickeln, Erstellen und Betreuen von IT-Lösungen

- systematisch Fehler erkennen, analysieren und beheben
- Algorithmen formulieren und Anwendungen in einer Programmiersprache erstellen
- Datenbankmodelle unterscheiden, Daten organisieren und speichern sowie Abfragen erstellen
- Fehler erkennen, analysieren und beheben
- Algorithmen formulieren und Programme entwickeln
- Datenbanken modellieren und erstellen

Durchführen und Dokumentieren von qualitätssichernden Maßnahmen

- Ursachen von Qualitätsmängeln systematisch feststellen, beseitigen und dokumentieren
- im Rahmen eines Verbesserungsprozesses die Zielerreichung kontrollieren, insbesondere einen Soll-Ist-Vergleich durchführen
- Methoden der Qualitätslenkung anwenden
- Methoden zur Messung der Zielerreichung im QM-Prozess kennen und anwenden

Umsetzen, Integrieren und Prüfen von Maßnahmen zur IT-Sicherheit und zum Datenschutz

- Bedrohungsszenarien erkennen und Schadenspotenziale unter Berücksichtigung wirtschaftlicher und technischer Kriterien einschätzen
- Kunden und Kundinnen im Hinblick auf Anforderungen an die IT-Sicherheit und an den Datenschutz beraten
- Wirksamkeit und Effizienz der umgesetzten Maßnahmen zur IT-Sicherheit und zum Datenschutz prüfen
- Schadenspotenziale von IT-Sicherheitsvorfällen einschätzen und Schäden verhindern können
- Präventive IT-Sicherheitsmaßnahmen für verschiedene Bedrohungsszenarien planen und umsetzen
- Ziele zur Entwicklung von IT-Sicherheitskriterien definieren
- Kunden zur IT-Sicherheit beraten
- IT-Sicherheitsmaßnahmen mit verschiedenen Tools überprüfen
- Technische organisatorische Maßnahmen (TOM) kontrollieren

Betreiben von IT-Systemen

- Netzwerkkonzepte für unterschiedliche Anwendungsgebiete unterscheiden
- Datenaustausch von vernetzten Systemen realisieren
- Verfügbarkeit und Ausfallwahrscheinlichkeiten analysieren und Lösungsvorschläge unterbreiten
- Maßnahmen zur präventiven Wartung und zur Störungsvermeidung einleiten und durchführen
- Störungsmeldungen aufnehmen und analysieren sowie Maßnahmen zur Störungsbeseitigung ergreifen
- Dokumentationen zielgruppengerecht und barrierefrei anfertigen, bereitstellen und pflegen, insbesondere technische Dokumentationen, System- sowie Benutzerdokumentationen
- Schichtenmodelle, z.B. OSI, TCP/IP benennen und zuordnen können
- Netzwerkkomponenten vergleichen und analysieren können
- Netzwerkkonzepte (-topologien, -Infrastrukturen) benennen und charakterisieren
- Peer 2 Peer bzw. Client-Server-Konzepte vergleichen und hinsichtlich ihres Einsatzes bewerten können
- Übertragungsprotokolle erläutern und zielgerichtet einsetzen können
- Standortübergreifende und -unabhängige Kommunikation situationsgerecht auswählen und einrichten können
- Netzwerkrelevante Dienste administrieren können
- Anwendungsdienste sicherstellen können
- Risiken identifizieren, Maßnahmen planen und Ausfallwahrscheinlichkeiten berücksichtigen
- Maßnahmen zur Sicherstellung des Betriebes beurteilen können
- Monitoringsysteme anwenden und Ergebnisse interpretieren können
- Monitoringergebnisse analysieren und korrektive Maßnahmen bestimmen können
- Erstellen und Erweitern von Handbüchern für Benutzer und Systembetreuer

Inbetriebnehmen von Speicherlösungen

- Sicherheitsmechanismen, insbesondere Zugriffsmöglichkeiten und -rechte, festlegen und implementieren
- Speicherlösungen, insbesondere Datenbanksysteme, integrieren
- Technische und organisatorische Maßnahmen (TOM)
- Möglichkeiten der physischen/hardwaretechnischen Absicherung benennen
- Möglichkeiten der softwaretechnischen Absicherung implementieren können

- Verschiedene Service- und Liefermodelle benennen und bedarfsorientiert auswählen können
- Daten heterogener Quellen zusammenführen können
- Netzwerkkomponenten und -Protokolle beschreiben können

Programmieren von Softwarelösungen

- Programmspezifikationen festlegen, Datenmodelle und Strukturen aus fachlichen Anforderungen ableiten sowie Schnittstellen festlegen
- Programmiersprachen auswählen und unterschiedliche Programmiersprachen anwenden
- Teilaufgaben von IT-Systemen automatisieren
- Anforderungen kundengerecht erfassen können
- Planen mit geeigneten Modellen
- Festlegen von Schnittstellen und vorhandene Schnittstellen nutzen
- Situationsgerechte Auswahl einer passenden Programmiersprache begründen können
- Algorithmen in einer Programmiersprache darstellen

- Die Darstellung soll in allgemein verständlichem Programm- oder Pseudocode erfolgen. Der Code soll für Dritte, ohne Kenntnis der verwendeten Programmiersprache, lesbar sein. Der Code muss nicht in der geschriebenen Sprache kompilierbar bzw. ausführbar sein.

Konzipieren und Umsetzen von kundenspezifischen Softwareanwendungen

- Vorgehensmodelle und -methoden sowie Entwicklungsumgebungen und -bibliotheken auswählen und einsetzen
- Analyse- und Designverfahren anwenden
- Benutzerschnittstellen ergonomisch gestalten und an Kundenanforderungen anpassen Anwendungslösungen unter Berücksichtigung der bestehenden Systemarchitektur entwerfen und realisieren
- bestehende Anwendungslösungen anpassen
- Datenaustausch zwischen Systemen realisieren und unterschiedliche Datenquellen nutzen
- komplexe Abfragen aus unterschiedlichen Datenquellen durchführen und Datenbestandsberichte erstellen
- Vorgehensmodelle unterscheiden können
- Strukturierte Analyse- und Designverfahren anwenden können

- Objektorientierte Analyse- und Designverfahren anwenden können
- Programmspezifikationen festlegen, Datenmodelle und Strukturen aus fachlichen Anforderungen ableiten, Schnittstellen festlegen, geeignete Programmiersprachen auswählen
- Konzepte von Programmiersprachen (z. B. strukturiert, prozedural, funktional, objektorientiert) kennen und exemplarisch Programmiersprachen nennen können
- Software-Entwicklungswerkzeuge aufgabenbezogen anwenden können
- Einsatzmöglichkeiten von Programmiersprachen kennen
- Lasten-/Pflichtenheft erstellen können
- UML-Diagramme erstellen können
- Datenmodelle erstellen können
- Normalisierung anwenden können (1. bis 3. Normalform)
- Design-Pattern anwenden können
- Anforderungen an die Softwareergonomie benennen und beurteilen können
- Benutzeroberfläche gestalten können
- Prototypen (Mock-ups) erstellen können
- Algorithmen erstellen können
- Objektorientierte Programmiermethoden anwenden können
- Einfache Such- und Sortier-Algorithmen kennen
- Bestehende Funktionen/Klassen erweitern
- Dateiformate zum Datenaustausch anwenden können und deren Einsatzbereiche kennen
- Möglichkeiten zur Nutzung von Services und Ressourcen eines Servers kennen
- Datenbankabfrage, Datenpflege mit SQL erstellen können

Sicherstellen der Qualität von Softwareanwendungen

- Sicherheitsaspekte bei der Entwicklung von Softwareanwendungen berücksichtigen
- Datenintegrität mithilfe von Werkzeugen sicherstellen
- Modultests erstellen und durchführen
- Werkzeuge zur Versionsverwaltung einsetzen
- Testkonzepte erstellen und Tests durchführen sowie Testergebnisse bewerten und dokumentieren
- Daten und Sachverhalte aus Tests multimedial aufbereiten und situationsgerecht unter Nutzung digitaler Werkzeuge und unter Beachtung der betrieblichen Vorgaben präsentieren
- Anwendungen unter Berücksichtigung von Datenschutz und Datensicherheit erstellen können

- Datenintegrität mithilfe von technischen Maßnahmen beurteilen und sicherstellen können
- Modultests erstellen und durchführen können (Unit-Tests)
- Grundfunktionalitäten einer Versionsverwaltung in ihrem Einsatz beschreiben und anwenden können, z. B. Branches, Pull, Push, Merge
- Softwaretests erstellen, durchführen und die Ergebnisse analysieren können
- Daten und Sachverhalte aus Tests multimedial aufbereiten und situationsgerecht unter Nutzung digitaler Werkzeuge und unter Beachtung der betrieblichen Vorgaben präsentieren

Themencluster

Die folgenden **Themencluster** sind konkrete Prüfungsinhalte aus dem Prüfungskatalog, der BIBB-Umsetzungshilfe* und echten IHK-Prüfungen, ergänzt um **meine eigenen** Empfehlungen/Stichpunkte. Die **Gruppierung** habe ich selbst vorgenommen, um sie auf konkrete Lernzielkontrollen zu verteilen.

Kundenbeziehungen

- Kundenbeziehungen unter Beachtung rechtlicher Regelungen und betrieblicher Grundsätze gestalten
- Kundengespräche strukturiert vorbereiten, durchführen und nachbereiten
- konsequente Kundenausrichtung und systematische Gestaltung der Kundenbeziehungsprozesse („Relationship Marketing“)
- Dokumentation und Verwaltung von Kundenbeziehungen („Customer Relationship Management“)
- Gesetz gegen unlauteren Wettbewerb
- AGB-Gesetz
- Regelkonformität
- Berücksichtigung der geltenden Compliance-Regelungen
- Ethik

Präsentieren

- Gespräche situationsgerecht führen und Kunden und Kundinnen unter Berücksichtigung der Kundeninteressen beraten
- Daten und Sachverhalte interpretieren, multimedial aufbereiten und situationsgerecht unter Nutzung digitaler Werkzeuge und unter Berücksichtigung der betrieblichen Vorgaben präsentieren

- Präsentieren von Sachverhalten (auch softwarebasiert) unter Berücksichtigung von z.B. Gestaltungsgrundsätzen nach Kundenvorgaben, Dateiformaten
- Anwenden von Kommunikations- und Argumentationstechniken
- Präsentationstechniken
- Grafische Darstellung bzw. Visualisierung (Diagrammarten, Bilderbearbeitung, Videos, multimediale Aufbereitung)
- Tabellenkalkulation
- Präsentationsprogramme
- Programme zum Erstellen multimedialer Inhalte
- Corporate Identity (CI)
- Anwendung und Einarbeitung in marktübliche Präsentationssoftware
- Vor- und Nachbereitung einer Präsentation
- Elemente einer Präsentation beherrschen, z.B.:

- Visualisierungsregeln

- Farbwirkung

- Rhetorikgrundlagen, z.B.:

- Atem- und Sprechtechnik

- Rede- und Vortragstechnik

Trends

- Technologische Entwicklungstrends von IT-Systemen feststellen sowie ihre wirtschaftlichen, sozialen und beruflichen Auswirkungen aufzeigen
- Veränderungen von Einsatzfeldern für IT-Systeme aufgrund technischer, wirtschaftlicher und gesellschaftlicher Entwicklungen feststellen
- Identifikation von Trends unter Berücksichtigung von Such- und Innovationsfeldern
- Beschaffen von Informationen über Auswirkungen auf das eigene Unternehmen, die Branche und die Gesellschaft
- Maßnahmen zur aktiven Information durch Newsfeeds oder Newsletter einleiten
- Nutzen geeigneter Informationsquellen, z.B. Fachmessen, Fachforen im Internet, um neue Trends und Einsatzfelder wahrzunehmen
- Anwendung von IT-Systemen auf neue Einsatzgebiete prüfen
- Ausfallsicherheit, bspw. redundante Systeme, selbstkonfigurierende Systeme
- Lebenslanges Lernen
- Teilhabe, soziale Stabilität
- Geräteklassen wie Tablets, Smartphones, Watches, Wearables

- Vernetzung, Integration und Modularisierung, Zentralisierung/Dezentralisierung, Embedded Systems
- Künstliche Intelligenz (KI), Machine Learning (ML), autonome Systeme

- Predictive Maintenance

- Big Data

- 3V-Modell (Velocity, Volume, Variety), 4V-Modell: +Veracity

- Streaming Analytics

- Cloud Computing

- Serverless (Function as a Service, Faas)

- Augmented Reality, Virtual Reality
- Internet of Things (IoT), Industrie 4.0

- Smart Grid

- Reactive Programming
- Microservice-Architektur
- Apps

- nativ vs. hybrid vs. cross-platform vs. responsive Web

- Progressive Web Apps

- Blockchain, Smart Contracts, Crypto-Currency
- Container (Docker) und Kubernetes
- Low-Code-Plattformen

Datenbanken

- Datentypen: Boolesche Werte, Ganzzahl, Gleitkommawerte, Währung, Datumswerte, Texte fester und variabler Länge, BLOB, Geokoordinaten
- OpenData, API-Schnittstellen
- Berücksichtigung vorhandener Datenbank- und Speicherkonzepte bei der Integration und Erweiterung von Bestandssystemen
- Inbetriebnahme von Speicherlösungen und Integration von Datenbanksystemen
- Beachten von Schnittstellen zu weiteren Systemen
- Datenquellen: nicht nur relationelle und schemafreie Datenbanken wie MySQL, MsSQL und MongoDB, sondern auch z. B. Sensoren, CSV-Dateien

- Begriffe kennen und erläutern

- Redundanz

- Kardinalitäten: 1:1, 1:n, m:n

- Primär-/Fremdschlüssel und andere Schlüsseltypen: anonym, künstlich/natürlich

- referentielle Integrität (Aktualisierungsweitergabe, Löschweitergabe)

- Maßnahmen bei Löschoperationen (Constraints): CASCADE, DENY/RESTRICT, SET NULL, (NO ACTION)

- Tiefergehende Datenbankobjekte: Index, Stored Procedure, Trigger, Sequence

- Replikation

- ACID-Prinzipien für Transaktionen kennen und erläutern (atomicity, consistency, isolation, durability)

Datenbankmodelle und -modellierung

- Datenbankmodelle unterscheiden, Daten organisieren und speichern sowie Abfragen erstellen
- Basiswissen zu verschiedenen Datenbankarchitekturen
- Relationale und nicht-relationale Datenbanken

- verschiedene Datenbankmodelle, z.B. hierarchisches Modell, Entity-Relationship-Modell, semantische Datenmodelle, objektorientierte Datenmodelle, als theoretische Grundlage für eine Datenbank kennen und nach Einsatzszenario unterscheiden

- NoSQL: dokumentenorientiert, spaltenorientiert, Key/Value-Store, objektorientiert, Graphendatenbank

- CAP-Theorem (Consistency, Availability, Partition Tolerance)

- Map/Reduce

- BASE (Basically Available, Soft State, Eventual Consistency)

- Phasen der Datenbankentwicklung kennen und anwenden

- externe Phase (Informationsbeschaffung)

- konzeptionelle Phase (Semantisches Modell)

- logische Phase (Datenmodell)

- physische Phase (Datenbankschema)

- Grundlagen der Datenmodellierung anwenden, z.B. Entitäten, Relationsbeziehungen, Normalisierung, Identifikationsschlüssel

- Definieren und Modellieren von Datenbankstrukturen, z.B. Entity-Relationship-Modell, Normalisierung

- ER-Diagramm (Entity Relationship Model): Entitätstypen, Attribute, Beziehungen, Kardinalitäten

- Crow’s Foot Notation

Normalisierung

- Normalformen erläutern („the key, the whole key, and nothing but the key“)

- Normalisierung von Datenbanken bis zur 3. Normalform durchführen

- Anomalien (Einfüge-, Änderungs-, Löschanomalie) erläutern
- Modellierung von Beziehungen (1:1, 1:n, m:n)

- mögliche Beispiele: Benutzer/Login (1:1), Benutzer/Bestellung (1:n), Benutzer/Benutzergruppe (m:n)

SQL

- Erstellen einfacher Abfragen von Datenquellen unter Verwendung einer Abfragesprache und komplexe Abfragen aus unterschiedlichen Datenquellen durchführen und Datenbestandsberichte erstellen
- SQL als normierte Sprache für die weit verbreiteten relationalen Datenbanken zum Bearbeiten (Einfügen, Verändern, Löschen) und Abfragen von darauf basierenden Datenbeständen anwenden
- Projektion vs. Selektion
- Kreuzprodukt/kartesisches Produkt
- DDL, DML, DQL, DCL, TCL
- CRUD (Create, Read, Update, Delete): INSERT/SELECT/UPDATE/DELETE
- SELECT-Aufbau rauf und runter: FROM, WHERE, JOIN, GROUP BY, HAVING, ORDER BY, (LIMIT)
- Subqueries
- LIKE-Syntax (Platzhalter)

- Abfrage über mehrere Tabellen (JOIN)

- verschiedene Joins erklären (INNER, OUTER LEFT/RIGHT/FULL, Natural, Self)

- Ausdrücke und Bedingungen
- Nutzung von Aggregatsfunktionen, z.B. COUNT, SUM, AVG
- Tabellenstruktur (CREATE, ALTER, DROP, DESCRIBE, SHOW DATABASES, SHOW TABLES)
- Manipulation (INSERT, UPDATE, DELETE)
- Projektion (SELECT FROM)
- Selektion (SELECT FROM… WHERE) und (SELECT… (SELECT…)), DISTINCT
- Sortieren (ORDER BY ASC/DESC)
- Gruppieren (GROUP BY, HAVING)
- Index (CREATE INDEX)
- Schnitt-, Vereinigungs- und Differenzmenge (INTERSECT, UNION (ALL), MINUS)
- SQL Injection

Qualitätssicherung

- Ursachen von Qualitätsmängeln systematisch feststellen, beseitigen und dokumentieren
- Verschiedene Prüfverfahren, z.B. Parität, Redundanz
- Debugging, Ablaufverfolgung
- Netzwerkanalyse, Bandbreite, Reaktionszeiten
- Im Rahmen eines Verbesserungsprozesses die Zielerreichung kontrollieren, insbesondere einen Soll-Ist-Vergleich durchführen
- Verbesserungsprozess, PDCA-Zyklus, KVP, Kennzahlen
- Soll-Ist-Vergleich, Abweichungen erkennen und berechnen
- Sicherheitsaspekte bei der Entwicklung von Softwareanwendungen berücksichtigen
- Datenintegrität mithilfe von Werkzeugen sicherstellen
- Constraints
- Validierungen
- Transaktionssicherheit
- übergeordneter Problemlösungsprozess

- Problemverständnis und -beschreibung (Define)

- Problemanalyse und Ursachensuche (Measure)

- Lösungssuche und -auswahl (Analyse)

- Lösungsrealisierung und -bewertung (Improve)

- Überprüfung der Wirksamkeit (Control)

- verschiedene Methoden, insbesondere in den Stadien „Ursachensuche“ und „Analysieren“, kennen und anwenden, z.B.:

- Ursachensuche: 6-W-Fragetechnik, Störungsmatrix, Histogramm, Verlaufsdiagramm, Korrelationsdiagramm

- Analysieren: Brainstroming/-writing, Flussdiagramm, Ishikawa-Diagramm, Variablenvergleich, Messsystemanalyse, Komponententausch, Einsatz von Debuggern

- Lösungsrealisierung bzw. Fehlerbehebung selbst vornehmen oder veranlassen und begleiten
- Grundlagen/Methoden des Qualitätsmanagements und einer vorbeugenden Qualitätssicherung bei IT-Systemen kennen und anwenden
- Qualitätsplanung (Ist-Zustand ermitteln und Ziel-Zustand festlegen)
- Qualitätslenkung (Umsetzung der Planphase)
- verschiedene Prüfverfahren kennen und bewerten, z. B. auf Parität, Redundanz
- Grundkenntnisse in der Stochastik (Berechnung von Wahrscheinlichkeiten bei Qualitätsmängeln)
- Qualitätssicherung (Auswertung relevanter Informationen)
- Qualitätsgewinn (weitere Umsetzung und Mitteilen der gewonnenen Informationen an die betroffenen Stellen)
- Qualitätsmanagement als selbstreferenziellen Prozess begreifen (die Verfahren zur Verbesserung lassen sich auch auf den Qualitätsmanagementprozess selbst anwenden)
- Erstellen und Erweitern von Handbüchern für Benutzer und Systembetreuer
- Berücksichtigung der Komplexität und Verständlichkeit bei der Nutzung von Herstellerdokumentationen zur Bereitstellung für den Anwender
- Incident Management (Ticketsystem)
- Standard Operation Procedures (SOP)
- Service Level Agreement (SLA), Service level 1 -3

Testen

- Klassifizierung von Testverfahren

- Wer testet?

  - Mensch (manuell) vs. Maschine (automatisch)

- Entwickler vs. Benutzer

- Was wird getestet?

- Komponente (Unit-Test/Funktionstest/Klassentest) vs. Integration vs. System (End-to-End)

- Testpyramide

- Wie wird getestet?

- Bottom-Up vs. Top-Down

- statisch (Kompilierzeit) vs. dynamisch (Laufzeit)

- ohne Kenntnis des Codes (Blackbox) vs. mit Kenntnis des Codes (Whitebox)

- explorativ

- Schreibtischtest/Review

- Wann wird getestet?

- Vor vs. nach der Entwicklung

- Abnahmetest

- Warum wird getestet?

- Regressionstest

- Lasttest/Belastungstest

- Smoketest

- Methoden zur Ermittlung von Testfällen

- Anweisungsüberdeckung vs. Zweig-/Pfadüberdeckung

- Äquivalenzklassen

- Grenzwertanalyse/Extremwertetest

- Modultests erstellen und durchführen

- Test-Doubles: Stubs vs. Mocks

- Eigenschaften guter Unit-Tests: korrekt, isoliert, schnell, aussagekräftig, wartbar, einfach durchführbar

- Testkonzepte erstellen und Tests durchführen sowie Testergebnisse bewerten und dokumentieren

- Definition der Inhalte eines Tests, z.B. Testkonzepte, Testdaten, Testszenario

- Beschreiben des Testumfangs, z.B. Grenzbelastung, Stabilität

- Testdatengeneratoren

- Daten und Sachverhalte aus Tests multimedial aufbereiten und situationsgerecht unter Nutzung digitaler Werkzeuge und unter Beachtung der betrieblichen Vorgaben präsentieren
- Testprozess

- Auswahl des Testverfahrens

- Kriterien für Testergebnisse definieren

- Testdaten generieren und auswählen

- Testprotokoll und Auswertung

- Auswerten von Testergebnissen, z.B. Soll-Ist-Vergleich

- Testprotokolle

- Kontrollverfahren

- Hardwaretest, z.B. Wareneingangskontrolle, mangelhafte Lieferung, Warenausgangskontrolle, Abnahmeprotokoll

- Software-Test, z.B. Testverfahren, Abnahmeprotokoll

Versionsverwaltung

- Werkzeuge zur Versionsverwaltung einsetzen
- Eigenschaften eines Versionsverwaltungssystems beschreiben

- SVN, CVS, TFS mit Source Safe, Git

- VCS vs. DVCS

- Nutzen und Anwenden einschlägiger Systeme, z.B. Git
- Funktionen, z.B. Commit, Revert, Branch, Merge, Cherry-Pick, Pull/Push, Rebase
- Übliche Workflows im Team, z.B. Pull/Merge Requests

IT-Sicherheit

- Datensicherheit (Authentifizierung, Autorisierung, Verschlüsselung)
- Bedrohungsszenarien erkennen und Schadenspotenziale unter Berücksichtigung wirtschaftlicher und technischer Kriterien einschätzen
- Für jede Anwendung, die verwendeten IT-Systeme und die verarbeiteten Informationen gilt: Betrachtung zu erwartender Schäden, die bei einer Beeinträchtigung von Vertraulichkeit, Integrität oder Verfügbarkeit entstehen könnten!

- Imageschaden

- Wirtschaftlicher Schaden

- Datenverlust

- Bedrohungsszenarien

- Datendiebstahl

- Digitale Erpressung (Ransomware)

- Identitätsdiebstahl (Phishing)

- Sicherheitskriterien

- Richtschnur für Entwickler

- Objektive Bewertung der Systeme (IT-Grundschutzmodellierung)

- Anwender/Benutzer bei der Auswahl eines geeigneten IT-Sicherheitsprodukts unterstützen (Security by Design)

- Kunden und Kundinnen im Hinblick auf Anforderungen an die IT-Sicherheit und an den Datenschutz beraten

- Private Haushalte

- Unternehmen (intern, extern)

- Öffentliche Hand

- Funktionale Anforderungen

- Qualitätsanforderungen Anforderungen

- Rahmenbedingungen

  - Technologisch

- Organisatorisch

- Rechtlich

- Ethisch

- Risikoanalyse

- Wirksamkeit und Effizienz der umgesetzten Maßnahmen zur IT-Sicherheit und zum Datenschutz prüfen
- Device Security Check
- Identity & Access Management (IAM)
- Schwachstellenanalyse (z.B. Ende-zu-Ende-Verschlüsselung)
- Zutritt vs. Zugang vs. Zugriff

- Zutrittskontrolle, z.B. Alarmanlage, Videoüberwachung, Besucherausweise

- Zugangskontrolle, z.B. Bildschirmschoner mit Passwortschutz, Biometrische Verfahren, Magnet- oder Chipkarte

- Zugriffskontrolle, z.B. Verschlüsselung von Datenträgern, Löschung von Datenträgern, User/Rollenkonzept

- Log Management
- Compliance Reports
- unterschiedliche Gefahrenquellen, z.B. Stromausfall, Überhitzung, Virenbefall
- geeignete Gegenmaßnahmen, z.B. USV-Anlagen, Klimageräte, Firewalls
- Einteilung in die drei Schutzbedarfskategorien „normal“, „hoch“ und „sehr hoch“ (analog IT-Grundschutz des BSI)
- IT-Sicherheitsregeln
- verschiedene IT-Sicherheitszertifizierungen
- Bundesamt für Sicherheit in der Informationstechnik (BSI) als Informationsplattform
- Basis-Sicherheitscheck für schnellen Überblick über das vorhandene IT-Sicherheitsniveau, z.B. als Soll/Ist-Abgleich der noch fehlenden Maßnahmen oder Interviews über den Status quo eines bestehenden Informationsverbundes
- ergänzende Sicherheitsanalyse mit Risikoanalyse (BSI-Standards 100-3)
- Sicherheitstest einzelner Rechner oder Netzwerke jeglicher Größe, z.B. durch Penetrationstest (auch Social-Engineering-Penetrationstest) gem. Klassifikationsschema des BSI
- Durchführung in einem fünfstufigen Prozess

- Vorbereitungsphase

- Informationsbeschaffung

- Bewertung der Informationen

- Versuch des aktiven Eindringens

- Auswertung der Ergebnisse

- mögliche Software, Portscanner, Sniffer, Paketgeneratoren, Passwortcracker, Verbindungsinterceptoren, Vulnerability Scanner etc. (siehe auch Open Vulnerability Assessment System – OpenVAS – unterstützt durch das BSI)
- Begriffe kennen/erläutern

- Hacker (White Hat, Black Hat), Cracker, Script-Kiddies

- Spam, Phishing, Sniffing, Spoofing, Man-in-the-Middle

- SQL-Injection, XSS, CSRF, Session Hijacking, DoS, DDoS

- https://xkcd.com/327/

- Viren, Würmer, Trojaner, Hoax, Dialer (veraltet), Keylogger, Botnetze, Spyware, Adware, Ransomware, Scareware

- Backdoor, Exploit, 0-Day-Exploit, Rootkit

- Verbreitung von Viren/Würmer/Trojaner erläutern

- Maßnahmen zur Angriffserkennung, z.B. Monitoring, Honeypot
- OWASP Top 10: Injection, Misconfiguration, Broken Access Control, Monitoring Failures usw.

- SQL Injection, Cross-Site-Scripting (XSS), Cross-Site-Request-Forgery (CSRF)

- Gegenmaßnahmen auf Entwicklerseite (z.B. Validierung, Cross-Origin-Resource-Sharing (CORS))

- Authentifizierungs- und Autorisierungsverfahren

- Delegierte Authentifizierung

- OAuth2

- Single-Sign-On

Datenschutz

- Datenschutzgesetze – national und auf EU-Ebene, z.B. Datenschutzgrundverordnung (DSGVO), BDSG

- Definition von personenbezogenen Daten

- Grundsätze des Datenschutzes (Art. 5)

- Rechtmäßigkeit/Gesetzmässigkeit (Erfordernis der gesetzlichen Grundlage)

- Transparenz gegenüber den betroffenen Personen

- Zweckbindung

- Datenminimierung/Verhältnismässigkeit (Datensparsamkeit und Datenvermeidung)

- Richtigkeit

- Speicherbegrenzung

- Integrität und Vertraulichkeit

- Rechenschaftspflicht

- Informationssicherheit

- Betroffenenrechte

- Recht auf Information

- Recht auf Auskunft

- Recht auf Berichtigung

- Recht auf Löschung

- Recht auf Einschränkung der Bearbeitung

- Recht auf Widerspruch

- Recht auf Datenübertragbarkeit

- Persönlichkeitsrechte

- Recht auf informationelle Selbstbestimmung

- Recht am eigenen Bild

- Recht am geschriebenen/gesprochenen Wort

- Recht auf Schutz vor Imitation der Persönlichkeit

- Recht auf Schutz der Intim-, Privat- und Geheimsphäre

- Archivierung (rechtliche Vorgaben, Unterschied zu Backup, technologische Anforderungen)

- Systeme, Fristen, Pflichten

Netzwerktechnik

- Netzwerkkonzepte für unterschiedliche Anwendungsgebiete unterscheiden
- Datenaustausch von vernetzten Systemen realisieren
- Verfügbarkeit und Ausfallwahrscheinlichkeiten analysieren und Lösungsvorschläge unterbreiten
- Maßnahmen zur präventiven Wartung und zur Störungsvermeidung einleiten und durchführen
- Störungsmeldungen aufnehmen und analysieren sowie Maßnahmen zur Störungsbeseitigung ergreifen
- Dokumentationen zielgruppengerecht und barrierefrei anfertigen, bereitstellen und pflegen, insbesondere technische Dokumentationen, System- sowie Benutzerdokumentationen
- Auswerten, Dokumentieren und Weiterleiten von Informationen und Störungsmeldungen
- Ergreifen von Maßnahmen zur Problembeseitigung und ggf. fachlicher Austausch mit Systemlieferanten
- ggf. Weiterleitung zur jeweiligen Fachabteilung oder Systemspezialisten
- Adressierung

- IPv4/IPv6, MAC, ARP

- Routing, Switching
- DNS, DHCP
- TCP/UDP
- HTTPS, TLS/SSL, IPsec

- Hash, Signatur, Zertifikat, Certificate Authority

- SMB, NFS
- Ethernet, FibreChannel
- Datenübertragungsraten
- Verschlüsselung (pre-shared key, RADIUS …)
- LAN/WAN/MAN/GAN
- Strukturierte Verkabelung

- primäre/sekundäre/tertiäre Verkabelung

- Kabeltypen

- Simplex, Halb-/Vollduplex

- 10/100/1000Base-T

- Twisted Pair, CAT5e/6/7 etc.

- Fibre Channel, Lichtwellenleiter

- DIN EN 50173-1

- EM-Verträglichkeit

- VLAN
- Drahtlos: PAN/WLAN, Bluetooth
- Sicherheitskonzepte und -risiken: WEP, WPA
- Netzwerktopologien
- Netzwerkplan
- VPN

- Funktionsweise und Vorteile von VPN beschreiben

- Protokolle/Ports, Verschlüsselungsverfahren

- L2TP, PPTP, IPSec

- VPN-Modelle

- Tunneling

- Echtzeitkommunikation sicherstellen können
- Serverarten: Mailserver, Webserver, Groupware, Datenbanken, Proxy
- ANR
- Notfallkonzept (Disaster Recovery)
- Sicherstellung des Betriebs

- Elektrotechnisch (USV)

- Hardwaretechnisch (Redundanzen), RAID

- Softwaretechnisch (Back-ups…)

- MTBF

- SNMP, S.M.A.R.T. u.Ä.
- Systemlastanalyse
- Predictive Maintenance

- Clustering, Load Balancing

- Round Robin

- Firewalls/Webfilter

Paketfilter, Stateful Inspection, Application Level

- Portsecurity, Port-Forwarding

Speicherlösungen

- Sicherheitsmechanismen, insbesondere Zugriffsmöglichkeiten und -rechte, festlegen und implementieren
- Speicherlösungen, insbesondere Datenbanksysteme, integrieren
- Zugangskontrollen (z.B. Gebäude, Serverraum, Schrank)
- Implementierung und Inbetriebnahme des Zugriffs auf lokale und vernetzte Speicherlösungen sowie vernetzten Systemen, z.B. SAN, NAS, DAS
- Berücksichtigung der Organisationsstrukturen im Unternehmen unter Beachtung von örtlichen Vorgaben
- Usermanagement
- Verschlüsselung (TPM)
- Fog, Cloud
- SaaS, XaaS
- Data Warehouse
- Data Lake

Softwareentwicklung [

- Arten von Software unterscheiden (Individual-/Branchensoftware)

- ERP, CRM, CAD, CMS, DMS, PPS, ECM

- Programmspezifikationen festlegen, Datenmodelle und Strukturen aus fachlichen Anforderungen ableiten sowie Schnittstellen festlegen
- Programmiersprachen auswählen und unterschiedliche Programmiersprachen anwenden
- Teilaufgaben von IT-Systemen automatisieren
- Datenbankverbindung implementieren
- Skript- und Shellprogrammierung (z.B. Python)

- herstellerabhängige Skriptbausteine und -sprachen anwenden, z.B.: Bash, PowerShell

- wiederkehrende Systemabläufe automatisieren und überwachen

- Optimieren und Automatisieren lokaler und netzwerkübergreifender Aufgaben

- Bestehende Anwendungslösungen anpassen
- Datenaustausch zwischen Systemen realisieren und unterschiedliche Datenquellen nutzen
- Allgemeines Fehlerhandling in Programmen

- Exceptions, Return/Exit Codes

- Unterschied syntaktische/semantische Fehler

- Systematisch Fehler erkennen, analysieren und beheben

- Debugging, Break Point

- Berücksichtigung anwendungsspezifischer Möglichkeiten, z.B. Makrosprache
- Rechnerarchitektur: CPU, BUS, Speicher und deren Adressierung
- Lizenzen unterscheiden

- Open Source, proprietär

- Informationspflichten zu Produkten, Namens- und Markenrecht, Urheber- und Nutzungsrecht, Persönlichkeitsrecht, unlauterer Wettbewerb

Algorithmen

- Algorithmus: präzise (eigentlich von IT-Systemen unabhängige) Formulierung einer Verarbeitungsvorschrift
- Algorithmen formulieren und Anwendungen in einer Programmiersprache erstellen
- Abbildung der Kontrollstrukturen mittels Struktogramm, PAP oder Pseudocode als didaktisches Hilfsmittel
- grundlegende Algorithmen kennen, eigene Algorithmen auch programmiersprachenfrei formulieren und zur Lösung von Problemen, z.B. in einem IT-System bzw. einer Softwareanwendung einsetzen
- Entwickeln und Darstellen von Programmlogiken unabhängig von der Programmiersprache, z.B. mithilfe von Struktogrammen nach Nassi-Shneidermann sowie Strukturdiagrammen und Verhaltensdiagrammen aus der UML
- Kontrollstrukturen

- allgemeine Programmstrukturen identifizieren/erläutern (Verzweigungen, Schleifen etc.)

- Merkmale/Unterschiede von Kontrollstrukturen (Schleifen, Fallunterscheidungen)

- grundlegende Kontrollstrukturen in allen Diagrammformen darstellen können: Pseudocode, Struktogramm/Nassi-Shneiderman, Programmablaufplan (PAP)

- Zustandsübergänge eines Zustandsautomaten abbilden

- Rekursion: Funktionsweise, Vor-/Nachteile
- Algorithmen implementieren/durchspielen

- Mittelwert

- doppelte Einträge in einem Array finden/löschen

- Dateibäume rekursiv kopieren

- (Zinses-)Zinsberechnung

- Planen eines regelmäßigen Backups

- Ablauf einer Benutzerauthentifizierung an einer Website

- Abbuchen von einem Konto

- Lineare Suche

- Binäre Suche

- Bubble Sort

- Reguläre Ausdrücke zur Textanalyse erstellen

Schnittstellen, APIs, Datenaustausch

- Datenaustauschformate: CSV, XML, JSON
- XML

- vs. SGML, HTML, CSV, JSON, YAML etc.

- Wohlgeformtheit, Validität

- Parser, Serialisierer

  - SAX, DOM

- DTD, Schema, RelaxNG, Schematron

- XSLT, XSL-FO

- JSON

- Syntax, Vor-/Nachteile, Einsatzgebiete

- REST

- Adressierbarkeit, Zustandslosigkeit, einheitliche Schnittstelle (uniform interface), Ressource vs. Repräsentation

- Hypermedia as the Engine of Application State (HATEOAS)

- Code on Demand

- Webservices/SOA

- SOAP, WSDL

Objektorientierung

- Prinzipien der OOP

- Begriffe der OOP erläutern: Attribut, Nachricht/Methodenaufruf, Persistenz, Schnittstelle/API/Interface, Polymorphie, Vererbung

- Bestandteile von Klassen

- Unterschied Klasse/Objekt

- Unterschied Klasse/Interface

- Erklärung Klassenbibliothek vs. Framework

- Klassenbeziehungen: Assoziation, Aggregation, Komposition, Spezialisierung, Generalisierung

- Generische Klassen (z.B. List)

- Vorteile generischer Container (Templates in C++) gegenüber Arrays

- Unterschied statische/nicht-statische Methoden und Attribute
- Datenstrukturen

- Queue, Baum, Stack, Heap, Array, Graph

- funktionale Aspekte in modernen Sprachen: Lambda-Ausdrücke, Functional Interfaces, Map/Filter/Reduce, deklarativ vs. imperativ

Programmiersprachen

- Programmierparadigmen: unstrukturiert, strukturiert, prozedural, funktional, objektorientiert, logisch
- Programmiersprachen vergleichen

- Compiler vs. Interpreter

- Programmierparadigma

- Typisierung: statisch/dynamisch, stark/schwach

- Syntax: z.B. C-ähnlich

- General Purpose vs. Domain Specific

- Abstraktionsniveau: 1GL, 2GL, 3GL, 4GL

- imperativ vs. deklarativ

- Portabilität (z.B. hardwarenah vs. virtuelle Maschine)

- Skriptsprache

- Performance/Speicherbedarf (Unterschiede bei der Programmierung/Ausführungsgeschwindigkeit in C, Java und JavaScript)

- Einsatzzweck(e)

- gängige Programmiersprachen kennen: PHP, Perl, Java, C, C++, C#, JavaScript, Delphi, Visual Basic, VBA, Ruby, Python, Cobol, F#, Lisp, Prolog, Assembler
- synchrone vs. asynchrone Programmierung
- Herausforderungen paralleler Programmierung
- Einsatz von integrierten Entwicklungsumgebungen (IDE)
- Framework vs. Bibliothek
- Eigenschaften funktionaler Programmierung

- Functions as „First Class Citizens“

- Pure Functions

- Higher Order Functions

- Immutability

- Fokus auf Rekursion (Tail Call Optimization)

- Pattern Matching

UML

- wichtige UML-Diagramme kennen und Einsatzgebiete erläutern
- Klassendiagramm
- Aktivitätsdiagramm
- Anwendungsfalldiagramm (Use-Cases)
- Sequenzdiagramm
- Zustandsdiagramm/Zustandsautomat
- Komponentendiagramm
- Verteilungsdiagramm

Softwarearchitektur

- Anwendungslösungen unter Berücksichtigung der bestehenden Systemarchitektur entwerfen und realisieren
- Berücksichtigung bestehender Systeme und Altsysteme
- Anpassung bzw. Weiterentwicklung bestehender Software an eine neue Umgebung
- Bottom-Up- und Top-Down-Verfahren bei der Modellierung erläutern
- Funktion/Vorteile der Modularisierung von Programmen
- Softwarearchitektur

- Monolith

- 3-Schichten-Modell/3-Tier

- Schichtenmodell/Layers

- Microservices

- Model View Controller (MVC)

- Model View Presenter (MVP)

- Model-View-ViewModel (MVVM)

- Pipes and Filters

- Webservice/Service Oriented Architecture (SOA)

- REST

- verteilte Anwendungen: Webservices, Microservices, Client-Server, Cloud
- Zustandslosigkeit, lose Kopplung

Softwareergonomie

- Mock-up
- Usability vs. User-Experience
- Entwurf der Bildschirmausgabemasken (Softwareergonomie, Barrierefreiheit)
- Benutzerschnittstellen ergonomisch gestalten und an Kundenanforderungen anpassen

- übliche Gestaltungselemente für Benutzerschnittstellen (Controls): Button, Textfeld, Dropdown, Combobox usw.

- Richtlinien bei der Gestaltung von Programmoberflächen

- Aufgabenangemessenheit

- Selbstbeschreibungsfähigkeit

- Lernförderlichkeit

- Steuerbarkeit

- Erwartungskonformität

- Individualisierbarkeit

- Fehlertoleranz (siehe Grundsätze der Dialoggestaltung)

- Barrierefreiheit bzw. Inklusives Design

Software Engineering

- Vorgehensmodelle und -methoden sowie Entwicklungsumgebungen und -bibliotheken auswählen und einsetzen
- Analyse- und Designverfahren anwenden
- Entwicklungsprozesse wie das Wasserfallmodell
- Iterative Modelle, z.B. Spiralmodell, V-Modell (XT)
- Agile Modelle: Scrum, Extreme Programming, Kanban
- Top-Down-Entwurf vs. Bottom-Up-Entwurf
- Entwicklungswerkzeuge: Editor, IDE, Programmgenerator, Linker, Compiler, Interpreter, Debugger, Testsoftware, Versionsverwaltung
- Erstellen von Spezifikationen von Daten- und Programmstrukturen auf angemessenem Abstraktionsniveau
- Nutzung von Prinzipien einer systematischen Programmierung nutzen (Strukturierung, Modularisierung, Mehrfachverwendung, Standardisierung)

- Vorteile von Modularisierung

- Anpassung aufgrund kundenspezifischer Anforderungen
- Anforderungen aufnehmen und dokumentieren

- funktionale/nicht-funktionale Anforderungen

- Eigen- vs. Fremdfertigung

Design Patterns

- Design Patterns kennen/erklären/implementieren

- Singleton, Observer, Factory, Adapter, Iterator, Strategy, Decorator, Template Method, Registry, MVC

Softwarequalität

- Beachten von Qualitätskriterien beim Programmieren mit branchentypischen Werkzeugen, Editoren, Entwicklungsumgebungen
- Anforderungen: Änderbarkeit, Benutzbarkeit, Effizienz, Funktionalität, Übertragbarkeit, Zuverlässigkeit, Normen anwenden
- Definition Software-Qualität
- Software-Qualitätsmerkmale nach ISO 9126 nennen und erläutern

- Funktionalität: Angemessenheit, Interoperabilität, Ordnungsmäßigkeit, Richtigkeit, Sicherheit

- Änderbarkeit: Analysierbarkeit, Modifizierbarkeit, Testbarkeit, Stabilität

- Übertragbarkeit: Anpassbarkeit, Austauschbarkeit, Installierbarkeit, Koexistenz

- Effizienz: Verbrauchsverhalten, Zeitverhalten

- Zuverlässigkeit: Fehlertoleranz, Reife, Wiederherstellbarkeit

- Benutzbarkeit: Attraktivität, Bedienbarkeit, Erlernbarkeit, Verständlichkeit

- Software-Qualitätsmerkmale nach ISO 25010 nennen und erläutern

- Functional Suitability: Functional Completeness, Functional Correctness, Functional Appropriateness

- Performance Efficiency: Time Behaviour, Resource Utilization, Capacity

- Compatibility: Co-existence, Interoperability

- Usability: Appropriateness Recognizability, Learability, Operability, User Error Protection, User Interface Aesthetics, Accessibility

- Reliability: Maturity, Availability, Fault Tolerance, Recoverability

- Security: Confidentiality, Integrity, Non-repudiation, Authenticity, Accountability

- Maintainability: Modularity, Reusability, Analysability, Modifiability, Testability

- Portability: Adaptability, Installability, Replaceability

- Maßnahmen zur Qualitätssicherung

- Audits, Code Reviews, Testmethoden, Entwicklungsprozess, Dokumentation, statische Codeanalyse, Pair Programming, Bugtracking

- Continuous Integration/Delivery/Deployment

Webentwicklung

- dynamische Websites (CGI, ASP, JSP, PHP)

- Applet und Servlet unterscheiden (veraltet)

- Web 2.0

- Social Networks, Wikis, Blogs, Twitter, Forum, Podcast

- Web 3.0
- RIA und AJAX

- Vor-/Nachteile

- Funktionsweise

- Anforderungen durch Mobilgeräte

- Offline-Fähigkeit, Deployment auf mehrere Plattformen, verschiedene Programmiersprachen, native Apps vs. HTML5/JavaScript, geringe Bandbreiten, kleine Auflösungen

- Angriffsmöglichkeiten gegen Anwendungen abgrenzen

- SQL-Injection, XSS, CSRF, Session Hijacking, DoS, DDoS

- HTTP

- Methoden kennen und einordnen: safe/sicher, idempotent

- Status-Codes kennen (z.B. 200, 404, 500)