Privater Masterstudiengang
Softwareentwicklung

tech technologische universität

tech technologische universität

Privater Masterstudiengang

Softwareentwicklung

- Modalität: **online**
- Dauer: **12 Monate**
- Qualifizierung: **TECH Technologische Universität**
- Aufwand: **16 Std./Woche**
- Zeitplan: **in Ihrem eigenen Tempo**
- Prüfungen: **online**

Internetzugang: www.techtitute.com/de/informatik/masterstudiengang/masterstudiengang-softwareentwicklung

Index

01

Präsentation

Um in einem der zukunftsträchtigsten Bereiche des IT-Sektors mitwirken zu können, muss der Experte wissenschaftlich und technologisch spezialisiert und darauf vorbereitet sein, die Herausforderungen, die sich in der Berufspraxis des Software-Engineering ergeben, effizient zu bewältigen. Dieser private Masterstudiengang zielt darauf ab, ein hohes Niveau der Beherrschung der Softwareentwicklung zu erreichen, und zwar durch die neuesten Fortschritte und Entwicklungen in diesem Bereich mit Hilfe einer Studienmethodik von maximaler Wirkung und außerordentlicher Flexibilität.

Eignen Sie sich in der aktuellsten Fortbildung auf dem Online-Bildungsmarkt die umfassendsten Kenntnisse im Software-Engineering an und arbeiten Sie an Entwicklungen in diesem dynamischen Berufsfeld mit"

Mit dem Fortschritt der neuen Technologien ist Software zu einem äußerst wichtigen Element in der heutigen Welt geworden. In den letzten Jahren hat sich gezeigt, dass es notwendig ist, Softwareprodukte mit den richtigen Funktionen und in der richtigen Qualität zu entwickeln und dabei den Zeit-und Kostenrahmen einzuhalten.

Dieses Programm richtet sich an diejenigen, die ein höheres Niveau an Kenntnissen in der Softwareentwicklung erreichen wollen. Das Hauptziel besteht darin, die Studenten in die Lage zu versetzen, die in diesem privaten Masterstudiengang erworbenen Kenntnisse in der realen Welt anzuwenden, und zwar in einem Arbeitsumfeld, das die Bedingungen, mit denen sie in Zukunft konfrontiert werden könnten, in strenger und realistischer Weise wiedergibt.

Nutzen Sie die Möglichkeit, diese Weiterbildung zu 100% online zu absolvieren, ohne auf Verpflichtungen verzichten zu müssen, und erleichtern Sie sich so die Rückkehr an die Universität. Aktualisieren Sie Ihr Wissen und erwerben Sie einen privaten Masterstudiengang, um sich persönlich und beruflich weiterzuentwickeln.

Sie erwerben umfassende Kenntnisse auf dem Gebiet des Software-Engineering, aber auch auf dem Gebiet der Informatik und der Computerstruktur, einschließlich der mathematischen, statistischen und physikalischen Grundlagen, die für das Ingenieurwesen unerlässlich sind.

Nutzen Sie die Gelegenheit und absolvieren Sie diese Weiterbildung zu 100% online, ohne auf Ihre Verpflichtungen verzichten zu müssen, und erleichtern Sie sich so die Rückkehr an die Universität. Bringen Sie Ihr Wissen auf den neuesten Stand und erwerben Sie Ihren eigenen Masterstudiengang, um persönlich und beruflich voranzukommen.

Dieser **Privater Masterstudiengang in Softwareentwicklung** enthält das vollständigste und aktuellste wissenschaftliche Programm auf dem Markt. Die hervorstechendsten Merkmale sind:

- Entwicklung von 100 simulierten Szenarien, die von Experten für Softwareentwicklung vorgestellt werden
- Seine anschaulichen, schematischen und äußerst praktischen Inhalte, mit denen sie konzipiert sind, liefern wissenschaftliche und praktische Informationen zur Softwareentwicklung
- Neues über die jüngsten Entwicklungen in der Softwareentwicklung
- Er enthält praktische Übungen, in denen der Selbstbewertungsprozess durchgeführt werden kann um das Lernen zu verbessern
- Interaktives Lernsystem auf der Grundlage der Fallmethode und ihre Anwendung in der Praxis
- Ergänzt wird dies durch theoretische Vorträge, Fragen an den Experten, Diskussionsforen zu kontroversen Themen und individuelle Reflexionsarbeit
- Verfügbarkeit von Inhalten von jedem festen oder tragbaren Gerät mit Internetanschluss

Dieses Programm ermöglicht es Ihnen, die grundlegende Struktur eines Computers und seiner Software kennenzulernen, um Ihre Fähigkeiten zu erweitern“

Lernen Sie alles, was Sie brauchen, um sicher mit Programmiersprachen zu arbeiten. Ergänzen Sie Ihr Wissen um die Interpretation und den Entwurf grundlegender Algorithmen für die Arbeit in der Programmierung"

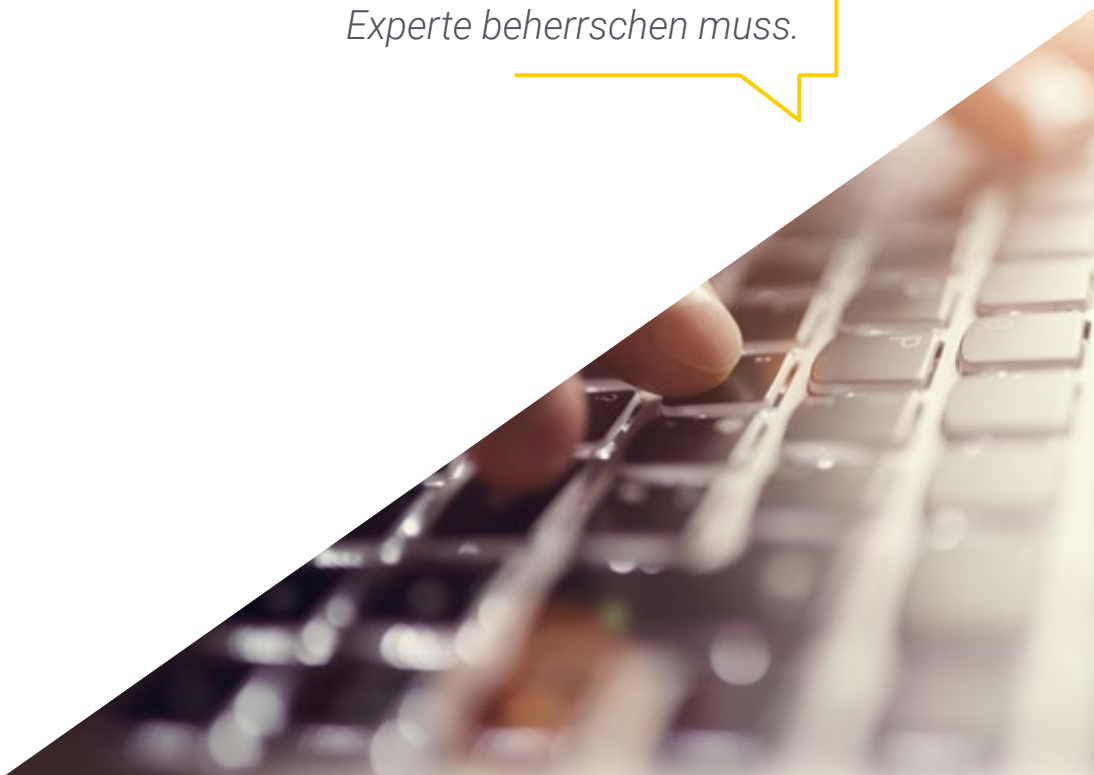
Das Lehrpersonal besteht aus Fachleuten aus der Welt der Softwareentwicklung, die ihre Berufserfahrung in diese Fortbildung einbringen, sowie aus anerkannten Fachleuten aus führenden Unternehmen und renommierten Universitäten.

Dank seiner multimedialen Inhalte, die mit den neuesten Bildungstechnologien entwickelt wurden, wird es den Fachleuten ermöglicht, in einer situierten und kontextbezogenen Weise zu lernen, d. h. in einer simulierten Umgebung, die ein immersives Lernen ermöglicht, das auf die Ausführung in realen Situationen programmiert ist.

Das Konzept dieses Programms konzentriert sich auf problemorientiertes Lernen, bei dem der Dozent versuchen muss, die verschiedenen Situationen aus der beruflichen Praxis zu lösen, die während des Programms gestellt werden. Dabei wird der Experte durch ein innovatives interaktives Videosystem unterstützt, das von anerkannten Experten für Softwareentwicklung mit umfassender Lehrerfahrung entwickelt wurde.

Eine Fortbildung, die Sie in die Lage versetzt zu verstehen, wie ein Computerprogramm funktioniert und wie Sie in alle wesentlichen Elemente eines Computerprogramms eingreifen können.

Lernen Sie die neuesten Datensysteme auf dem Markt kennen, erfahren Sie, wie man fortschrittliche Algorithmen entwirft und alle Aspekte, die ein hochkompetenter Experte beherrschen muss.



02

Ziele

Ziel dieser Fortbildung ist es, Fachleuten, die in der Softwareentwicklung tätig sind, die notwendigen Kenntnisse und Fähigkeiten zu vermitteln, um ihre Tätigkeit unter Verwendung der fortschrittlichsten Protokolle und Techniken ausüben zu können. Mit Hilfe eines Arbeitsansatzes, der vollständig an den Studenten angepasst werden kann, wird dieser private Masterstudiengang Sie schrittweise dazu bringen, die Fähigkeiten zu erwerben, die Sie auf ein höheres berufliches Niveau befördern werden.

Sie werden in das Gebiet der Berechnung und der Computerstruktur eintauchen, wesentliche Themen für jeden Softwareentwickler. für jeden Softwareentwickler"

Allgemeine Ziele

- Wissenschaftliche und technologische Spezialisierung sowie Vorbereitung auf die Berufspraxis des Software-Engineerings mit einer transversalen und vielseitigen Weiterbildung, die an die neuen Technologien und Innovationen in diesem Bereich angepasst ist
- Erwerb umfassender Kenntnisse auf dem Gebiet des Software-Engineering, aber auch auf dem Gebiet der Berechnung und der Computerstruktur, einschließlich der mathematischen, statistischen und physikalischen Grundlagen, die für das Ingenieurwesen wesentlich sind

Erreichen Sie den gewünschten Wissensstand und beherrschen Sie die Softwareentwicklung mit dieser hochkarätigen Fortbildung auf hohem Niveau"

Spezifische Ziele

Modul 1. Grundlagen der Programmierung

- Verständnis der grundlegenden Struktur eines Computers, von Software und allgemeinen Programmiersprachen
- Lernen Algorithmen zu entwerfen und zu interpretieren, die die notwendige Grundlage für die Softwareentwicklung sind
- Die wesentlichen Elemente eines Computerprogramms verstehen, z. B. die verschiedenen Datentypen, Operatoren, Ausdrücke, Anweisungen, E/A- und Steueranweisungen
- Verstehen der verschiedenen Datenstrukturen, die in allgemeinen Programmiersprachen zur Verfügung stehen, sowohl statisch als auch dynamisch, und Aneignung grundlegender Kenntnisse im Umgang mit Dateien
- Verstehen der verschiedenen Softwaretesttechniken und der Bedeutung der Erstellung einer guten Dokumentation zusammen mit einem guten Quellcode
- Die Grundlagen der Programmiersprache C++ kennenlernen, eine der am weitesten verbreiteten Programmiersprachen der Welt

Modul 2. Datenstruktur

- Die Grundlagen der Programmierung in der Sprache C++ lernen, einschließlich Klassen, Variablen, bedingte Ausdrücke und Objekte
- Abstrakte Datentypen, lineare Datenstrukturtypen, einfache und komplexe hierarchische Datenstrukturen und deren Implementierung in C++ verstehen
- Verstehen der Funktionsweise von fortgeschrittenen Datenstrukturen, die nicht den üblichen entsprechen
- Die Theorie und Praxis der Verwendung von Prioritätshügeln und Prioritätswarteschlangen verstehen
- Lernen wie *Hashtabellen* als abstrakte Datentypen und Funktionen funktionieren
- Verständnis der Graphentheorie sowie fortgeschrittener Graphalgorithmen und -konzepte

Modul 3. Algorithmus und Komplexität

- Die wichtigsten Strategien für den Entwurf von Algorithmen sowie die verschiedenen Methoden und Maßnahmen zur Berechnung von Algorithmen kennenlernen
- Kenntnis der wichtigsten Sortieralgorithmen, die in der Softwareentwicklung verwendet werden
- Verstehen, wie verschiedene Algorithmen mit Bäumen *Heaps* und Graphen arbeiten
- *Greedy-Algorithmen*, ihre Strategie und Beispiele für ihre Anwendung bei den wichtigsten bekannten Problemen verstehen. Die Anwendung von Greedy-Algorithmen auf Graphen werden wir ebenfalls kennenlernen
- Wir lernen die wichtigsten Strategien der Minimalpfadfindung kennen, mit dem Ansatz wesentlicher Probleme des Fachgebiets und Algorithmen zu deren Lösung
- Verstehen der Backtracking-Technik und ihrer wichtigsten Anwendungen sowie alternativer Techniken

Modul 4. Datenbanken

- Lernen Sie die verschiedenen Anwendungen und Zwecke von Datenbanksystemen sowie deren Betrieb und Architektur kennen
- Verstehen des relationalen Modells, von seiner Struktur und seinen Operationen bis hin zur erweiterten relationalen Algebra
- Lernen Sie, was SQL-Datenbanken sind, wie sie funktionieren, wie man Daten definiert und wie man Abfragen erstellt, von den einfachsten bis zu den fortgeschrittensten und komplexesten
- Lernen Sie, wie man Datenbanken mit Hilfe des Entity-Relationship-Modells entwirft, wie man Diagramme erstellt und die Eigenschaften des erweiterten E-R-Modells kennen
- Vertiefung des Entwurfs von relationalen Datenbanken, Analyse der verschiedenen Normalformen und Zerlegungsalgorithmen
- Schaffung der Grundlagen für das Verständnis der Funktionsweise von NoSQL-Datenbanken und Vorstellung der Datenbank MongoDB

Modul 5. Fortgeschrittene Datenbanken

- Vorstellung der verschiedenen Datenbanksysteme, die derzeit auf dem Markt erhältlich sind
- Erlernen der Verwendung von XML und Datenbanken für das Web
- Verstehen der Funktionsweise fortgeschrittener Datenbanken, wie z. B. parallele und verteilte Datenbanken
- Die Bedeutung von Indizierung und Assoziierung in Datenbanksystemen verstehen
- Die Funktionsweise von transaktionalen Verarbeitungs- und Abfragesystemen verstehen
- Erwerb von Kenntnissen in Bezug auf nicht-relationale Datenbanken und Data Mining

Modul 6. Fortgeschrittener Algorithmusentwurf

- Vertiefung des fortgeschrittenen Algorithmusentwurfs, Analyse von rekursiven und Divide-and-Conquer-Algorithmen sowie Durchführung von amortisierten Analysen
- Verstehen der Konzepte der dynamischen Programmierung und der Algorithmen für NP-Probleme
- Verstehen, wie kombinatorische Optimierung funktioniert, sowie die verschiedenen Zufallsalgorithmen und parallelen Algorithmen
- Wissen und verstehen, wie die verschiedenen Methoden der lokalen Suche und der Kandidatensuche funktionieren
- Lernen der Mechanismen der formalen Programmverifikation und der iterativen Programmverifikation kennen, einschließlich der Logik erster Ordnung und des formalen Systems von Hoare
- Lernen der Funktionsweise einiger der wichtigsten numerischen Methoden wie die Bisektionsmethode, die Newton-Raphson-Methode und die Sekantenmethode kennen

Modul 7. Mensch-Computer-Interaktion

- Aneignung solider Kenntnisse über die Interaktion zwischen Mensch und Computer und die Schaffung benutzbarer Schnittstellen
- Verstehen, wie wichtig die Benutzerfreundlichkeit von Anwendungen ist und warum es wichtig ist, sie bei der Entwicklung unserer Software zu berücksichtigen
- Die verschiedenen Arten menschlicher Vielfalt und die damit verbundenen Einschränkungen zu verstehen und zu wissen, wie man Schnittstellen an die spezifischen Bedürfnisse jeder Art von Vielfalt anpasst
- Den Prozess des Schnittstellendesigns kennenlernen, von der Anforderungsanalyse bis zur Evaluierung, über die verschiedenen Zwischenstufen, die zur Realisierung einer geeigneten Schnittstelle notwendig sind
- Kenntnis der verschiedenen Zugänglichkeitsrichtlinien, der Normen, die sie festlegen, und der Hilfsmittel, die es uns ermöglichen, sie zu bewerten
- Verstehen der verschiedenen Methoden der Interaktion mit dem Computer unter Verwendung von Peripheriegeräten und Geräten

Modul 8. Fortgeschrittene Programmierung

- Vertiefung der Kenntnisse in der Programmierung, insbesondere in Bezug auf die objektorientierte Programmierung und der verschiedenen Arten von Beziehungen zwischen bestehenden Klassen
- Kenntnis der verschiedenen Entwurfsmuster für objektorientierte Probleme
- Lernen der ereignisgesteuerten Programmierung und die Entwicklung von Benutzeroberflächen mit Qt kennen
- Aneignung der grundlegenden Kenntnisse über nebenläufige Programmierung, Prozesse und Threads
- Lernen, wie man die Verwendung von Threads und Synchronisation handhabt, sowie die Lösung gängiger Probleme bei der gleichzeitigen Programmierung
- Die Bedeutung von Dokumentation und Tests bei der Softwareentwicklung zu verstehen

Modul 9. Entwicklung vernetzter Anwendungen

- Kenntnis der Merkmale der Auszeichnungssprache HTML und ihrer Verwendung bei der Erstellung von Webseiten in Verbindung mit CSS-Stilvorlagen
- Den Umgang mit der browserorientierten Programmiersprache JavaScript lernen und einige ihrer wichtigsten Funktionen kennen
- Verstehen der Konzepte der komponentenorientierten Programmierung und der Komponentenarchitektur
- Lernen, wie man das *Bootstrap Frontend Framework* für die Gestaltung von Websites verwendet
- Die Struktur des Controller-View-Modells bei der Entwicklung von dynamischen Websites verstehen
- Kenntnis der dienstleistungsorientierten Architektur und der Grundlagen des HTTP-Protokolls

Modul 10. Softwareentwicklung

- Grundlagen der Softwareentwicklung und -modellierung, Erlernen der wichtigsten Prozesse und Konzepte
- Verstehen des Softwareprozesses und der verschiedenen Modelle für die Softwareentwicklung einschließlich agiler Technologien
- Verstehen von Requirements Engineering, deren Entwicklung, Ausarbeitung, Verhandlung und Validierung
- Lernen der Modellierung von Anforderungen und der verschiedenen Elemente wie Szenarien, Informationen, Analyseklassen, Fluss, Verhalten und Muster
- Verstehen der Konzepte und Prozesse des Softwaredesigns, Lernen der Designarchitektur und des komponentenbasierten und musterbasierten Designs
- Kenntnis der wichtigsten Normen in Bezug auf Softwarequalität und Projektmanagement

03

Kompetenzen

Nach Bestehen der Prüfungen des Privaten Masterstudiengangs in Softwareentwicklung wird die Fachkraft die notwendigen beruflichen Fähigkeiten erworben haben, um qualitativ hochwertige Arbeit zu leisten, und sie wird auch neue Fähigkeiten und Techniken erworben haben, die ihr helfen werden, die zuvor erworbenen Computerkenntnisse zu ergänzen.

Verbessern Sie Ihre Fähigkeiten in der Softwareentwicklung und erreichen Sie die nächste Stufe als Profi in diesem sich ständig weiterentwickelnden Bereich“

Allgemeine Kompetenz

» Antworten auf den aktuellen Bedarf im Bereich der Softwareentwicklung

Ein außergewöhnliches Programm in Bezug auf seine Dichte, seine Vollständigkeit und die Art und Weise, wie es gelehrt wird, das es Ihnen ermöglicht, schnell und effizient voranzukommen"

Spezifische Kompetenzen

- In der Lage sein, die grundlegende Struktur eines Computers, von Software und allgemeinen Programmiersprachen zu verstehen
- Wissen, wie man die Grundlagen der C++-Programmierung anwendet, einschließlich Klassen, Variablen, bedingte Ausdrücke und Objekte
- Vertiefte Kenntnisse der wichtigsten Strategien für den Entwurf von Algorithmen sowie der verschiedenen Methoden und Maßnahmen zu deren Berechnung
- Die verschiedenen Anwendungen und Zwecke von Datenbanksystemen sowie ihre Funktionsweise und Architektur verstehen und in der täglichen Praxis anwenden können
- In der Lage sein, die verschiedenen auf dem Markt befindlichen Datenbanksysteme vorzustellen
- Wissen, wie man rekursive und Divide-and-Conquer-Algorithmen analysiert und eine amortisierte Analyse durchführt
- Anwendung von Kenntnissen über die Interaktion zwischen Mensch und Computer und die Schaffung benutzbarer Schnittstellen in der täglichen Berufspraxis
- Vertiefte Kenntnisse in der Programmierung
- Kenntnis der Merkmale der Auszeichnungssprache HTML und ihrer Verwendung bei der Erstellung von Webseiten in Verbindung mit CSS-Stilvorlagen
- In der Lage sein, die wichtigsten Prozesse und Konzepte der Grundlagen des Software-Engineering und der Modellierung anzuwenden

04

Struktur und Inhalt

Die Struktur der Inhalte wurde von einem Team von Fachleuten aus dem Bereich der technischen Informatik entwickelt, um sicherzustellen, dass die Studenten des privaten Masterstudiengangs effizient und schnell lernen können. Zu diesem Zweck wurden die Inhalte so organisiert, dass ein intensives und konstantes Lernen möglich ist, wobei versucht wird, die Motivation aufrechtzuerhalten, die auf dem Gefühl des Lernfortschritts des Studenten beruht.

Ein Fortbildungsprogramm, das darauf abzielt, vollständige Softwareentwicklungsfähigkeiten zu erlangen, die Sie auf ein neues berufliches Niveau heben"

Modul 1. Grundlagen der Programmierung

1.1. Einführung in die Programmierung
- 1.1.1. Grundlegender Aufbau eines Computers
- 1.1.2. Software
- 1.1.3. Programmiersprachen
- 1.1.4. Lebenszyklus einer Softwareanwendung

1.2. Entwurf von Algorithmen
- 1.2.1. Lösung von Problemen
- 1.2.2. Deskriptive Techniken
- 1.2.3. Elemente und Struktur eines Algorithmus

1.3. Elemente eines Programms
- 1.3.1. Herkunft und Merkmale der Sprache C++
- 1.3.2. Das Entwicklungsumfeld
- 1.3.3. Programmkonzept
- 1.3.4. Grundlegende Datentypen
- 1.3.5. Betreiber
- 1.3.6. Ausdrücke
- 1.3.7. Programmiersätze
- 1.3.8. Dateneingabe und -ausgabe

1.4. Kontrollsätze
- 1.4.1. Programmiersätze
- 1.4.2. Abzweigungen
- 1.4.3. Bucles

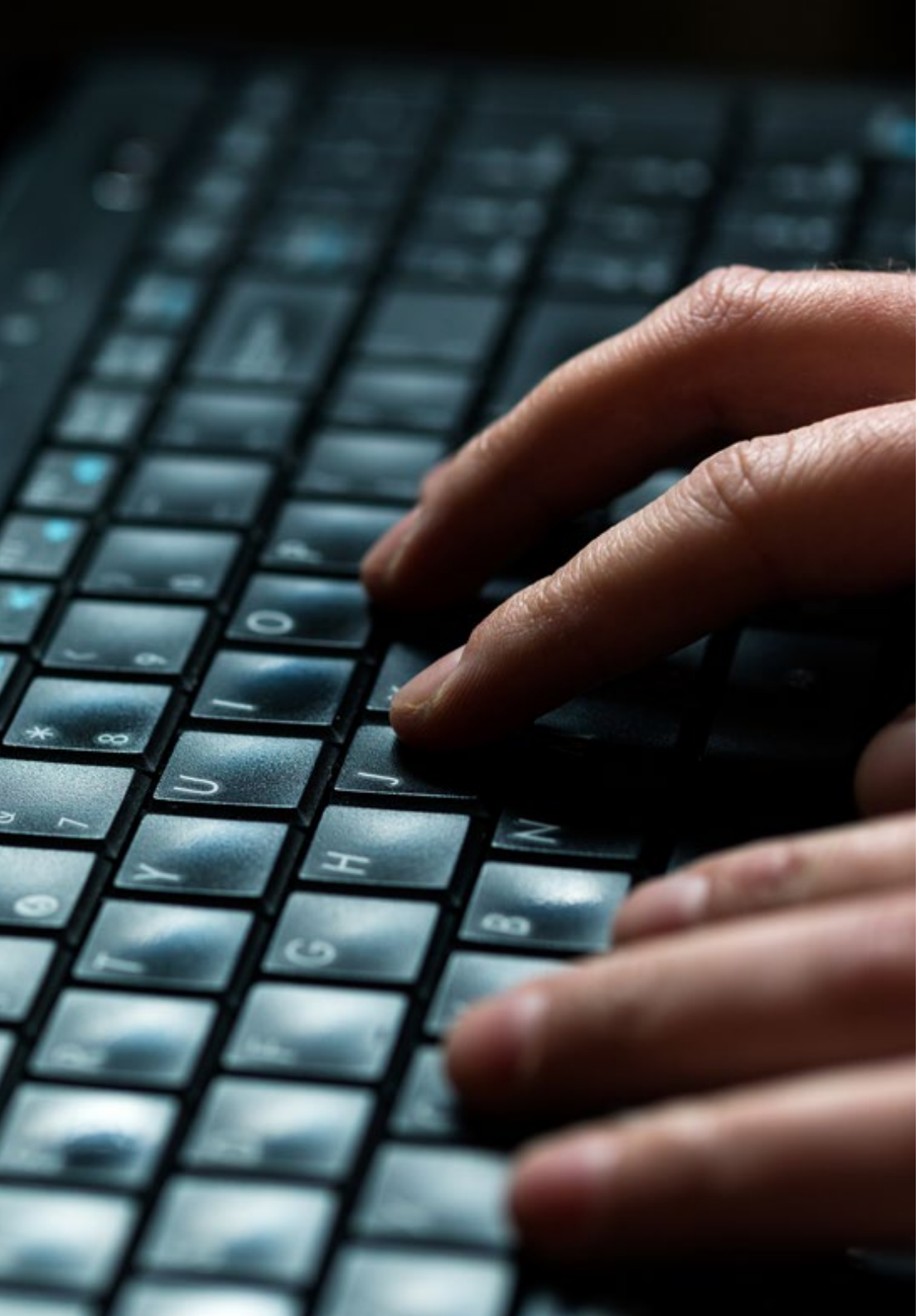
1.5. Abstraktion und Modularität: Funktionen
- 1.5.1. Modularer Aufbau
- 1.5.2. Begriff der Funktion und des Nutzens
- 1.5.3. Definition einer Funktion
- 1.5.4. Ausführungsablauf bei einem Funktionsaufruf
- 1.5.5. Prototyp einer Funktion
- 1.5.6. Rückgabe der Ergebnisse
- 1.5.7. Aufrufen einer Funktion: Parameter
- 1.5.8. Parameterübergabe durch Verweis und durch Wert
- 1.5.9. Identifizierungsbereich

1.6. Statische Datenstrukturen
- 1.6.1. *Arrays*
- 1.6.2. Matrizen Polyeder
- 1.6.3. Suchen und sortieren
- 1.6.4. Ketten. E/A-Funktionen für Zeichenketten
- 1.6.5. Strukturen Verbindungen
- 1.6.6. Neue Datentypen

1.7. Dynamische Datenstrukturen: Pointer
- 1.7.1. Konzept Definition von Pointer
- 1.7.2. Operatoren und Operationen mit Pointers
- 1.7.3. *Arrays* von Pointers
- 1.7.4. Pointers und *Arrays*
- 1.7.5. Pointers zu Zeichenketten
- 1.7.6. Pointers zu Strukturen
- 1.7.7. Mehrfache Unmittelbarkeit
- 1.7.8. Pointers zu Funktionen
- 1.7.9. Übergabe von Funktionen, Strukturen und *Arrays* als Funktionsparameter

1.8. Dateien
- 1.8.1. Grundlegende Konzepte
- 1.8.2. Dateioperationen
- 1.8.3. Dateitypen
- 1.8.4. Organisation der Dateien
- 1.8.5. Einführung in C++ Dateien
- 1.8.6. Handhabung von Dateien

1.9. Rekursion
- 1.9.1. Definition von Rekursion
- 1.9.2. Typen von Rekursionen
- 1.9.3. Vorteile und Nachteile
- 1.9.4. Überlegungen
- 1.9.5. Iterative rekursive Umwandlung
- 1.9.6. Der Rekursionsstapel



1.10. Tests und Dokumentation
- 1.10.1. Programmtests
- 1.10.2. White-Box-Test
- 1.10.3. Blackbox-Test
- 1.10.4. Prüfwerkzeuge
- 1.10.5. Programmdokumentation

Modul 2. Datenstruktur

2.1. Einführung in die Programmierung in C++
- 2.1.1. Klassen, Konstruktoren, Methoden und Attribute
- 2.1.2. Variablen
- 2.1.3. Bedingte Ausdrücke und Schleifen
- 2.1.4. Objekte

2.2. Abstrakte Datentypen (ADT)
- 2.2.1. Datentypen
- 2.2.2. Grundlegende Strukturen und ADTs
- 2.2.3. Vektoren und *Arrays*

2.3. Lineare Datenstrukturen
- 2.3.1. Definition der ADT-Liste
- 2.3.2. Verknüpfte und doppelt verknüpfte Listen
- 2.3.3. Geordnete Listen
- 2.3.4. Listen in C++
- 2.3.5. ADT-Stack
- 2.3.6. ADT-Queues
- 2.3.7. Stack und Queue in C++

2.4. Hierarchische Datenstrukturen
- 2.4.1. ADT-Baum
- 2.4.2. Patch
- 2.4.3. N-wertige Bäume
- 2.4.4. Binäre Bäume
- 2.4.5. Binäre Suchbäume

2.5. Hierarchische Datenstrukturen: komplexe Bäume
  2.5.1. Perfekt ausbalancierte Bäume oder Bäume mit Mindesthöhe
  2.5.2. Mehrpfadige Bäume
  2.5.3. Bibliografische Referenzen
2.6. Vorrangige Heaps und Queue
  2.6.1. ADT-Heaps
  2.6.2. ADT Vorrangige Queue
2.7. *Hash-Tabellen*
  2.7.1. ADT *Hash-Tabelle*
  2.7.2. *Hash-Funktionen*
  2.7.3. *Hash-Funktion* in *Hash-Tabellen*
  2.7.4. Redispersion
  2.7.5. Offene*Hash-Tabellen*
2.8. Graphs
  2.8.1. ADT-Graph
  2.8.2. Graph Typen
  2.8.3. Grafische Darstellung und Grundoperationen
  2.8.4. Graph Design
2.9. Fortgeschrittene Algorithmen und Konzepte für Graphs
  2.9.1. Probleme mit Graphs
  2.9.2. Algorithmen für Paths
  2.9.3. Suchalgorithmen oder -pfade
  2.9.4. Andere Algorithmen
2.10. Andere Datenstrukturen
  2.10.1. Sets
  2.10.2. Parallele *Arrays*
  2.10.3. Symboltabellen
  2.10.4. *Tries*

Modul 3. Algorithmus und Komplexität

3.1. Einführung in Algorithmenentwurfsstrategien
  3.1.1. Rekursion
  3.1.2. Aufteilen und erobern
  3.1.3. Andere Strategien
3.2. Effizienz und Analyse von Algorithmen
  3.2.1. Effizienzmassnahmen
  3.2.2. Messung der Größe der Eingabe
  3.2.3. Messung der Ausführungszeit
  3.2.4. Schlimmster Fall, bester Fall und mittlerer Fall
  3.2.5. Asymptotische Notation
  3.2.6. Mathematische Analysekriterien für nicht-rekursive Algorithmen
  3.2.7. Mathematische Analyse von rekursiven Algorithmen
  3.2.8. Empirische Analyse von Algorithmen
3.3. Sortieralgorithmen
  3.3.1. Sortier-Konzept
  3.3.2. Sortieren der Blase
  3.3.3. Sortierung nach Auswahl
  3.3.4. Sortierung durch Insertion
  3.3.5. Sortieren durch zusammenführen (Merge Sort)
  3.3.6. Schnelle Sortierung (Quick Sort)
3.4. Algorithmen mit Bäumen
  3.4.1. Baum-Konzept
  3.4.2. Binäre Bäume
  3.4.3. Baum-Patches
  3.4.4. Darstellung von Ausdrücken
  3.4.5. Geordnete binäre Bäume
  3.4.6. Ausgeglichene binäre Bäume
3.5. Algorithmen mit *Heaps*
  3.5.1. Die *Heaps*
  3.5.2. Der HeapSort Algorithmus
  3.5.3. Vorrangige Queues

3.6. Algorithmen mit Graphs
- 3.6.1. Vertretung
- 3.6.2. Patch in die Breite
- 3.6.3. Patch in die Tiefe
- 3.6.4. Topologische Ordnung

3.7. *Greedy* Algorithmen
- 3.7.1. Die *Greedy*-Strategie
- 3.7.2. Elemente der *Greedy*-Strategie
- 3.7.3. Währungsumtausch
- 3.7.4. Das Problem des Reisenden
- 3.7.5. Das Rucksack-Problem

3.8. Suche nach minimalen Pfaden
- 3.8.1. Das Problem des minimalen Pfades
- 3.8.2. Negative Bögen und Zyklen
- 3.8.3. Algorithmus von Dijkstra

3.9. Greedy Algorithmen auf Graphs
- 3.9.1. Der minimal überlappende Baum
- 3.9.2. Prims Algorithmus
- 3.9.3. Kruskals Algorithmus
- 3.9.4. Komplexitätsanalyse

3.10. *Backtracking*
- 3.10.1. Das *Backtracking*
- 3.10.2. Alternative Techniken

Modul 4. Datenbanken

4.1. Anwendungen und Zwecke von Datenbanksystemen
- 4.1.1. Anwendungen der verschiedenen Datenbanksysteme
- 4.1.2. Zweck in verschiedenen Datenbanksystemen
- 4.1.3. Übersicht der Daten

4.2. Datenbank und Architektur
- 4.2.1. Relationale Datenbanken
- 4.2.2. Datenbank-Design
- 4.2.3. Objektbasierte und semistrukturierte Datenbanken
- 4.2.4. Datenspeicherung und Abfragen
- 4.2.5. Transaktionsmanagement
- 4.2.6. Data Mining und Analyse
- 4.2.7. Datenbank-Architektur

4.3. Das relationale Modell: Struktur, Operationen und erweiterte relationale Algebra
- 4.3.1. Die Struktur von relationalen Datenbanken
- 4.3.2. Grundlegende Operationen in der relationalen Algebra
- 4.3.3. Andere Operationen der relationalen Algebra
- 4.3.4. Erweiterte Operationen der relationalen Algebra
- 4.3.5. Nullwerte
- 4.3.6. Modifizierung der Datenbank

4.4. SQL (I)
- 4.4.1. Was ist SQL?
- 4.4.2. Definition der Daten
- 4.4.3. Grundlegende Struktur der SQL-Abfragen
- 4.4.4. Operationen auf Sets
- 4.4.5. Aggregationsfunktionen
- 4.4.6. Nullwerte

4.5. SQL (II)
- 4.5.1. Verschachtelte Sub-Abfragen
- 4.5.2. Komplexe Abfragen
- 4.5.3. Ansichten
- 4.5.4. Cursors
- 4.5.5. Komplexe Abfragen
- 4.5.6. Triggers

4.6. Datenbankdesign und das ER-Modell
- 4.6.1. Überblick über den Entwurfsprozess
- 4.6.2. Das Entity-Relationship-Modell
- 4.6.3. Beschränkungen

4.7. Entity-Relationship-Diagramme
- 4.7.1. Entity-Relationship-Diagramme
- 4.7.2. Aspekte der Gestaltung von Entity-Beziehungen
- 4.7.3. Gruppen von schwachen Entitäten

4.8. Das erweiterte Entity-Relationship-Modell
- 4.8.1. Merkmale des erweiterten ER-Modells
- 4.8.2. Datenbank-Design
- 4.8.3. Reduktion auf relationale Schemata

4.9. Relationaler Datenbankentwurf
- 4.9.1. Merkmale eines guten relationalen Designs
- 4.9.2. Atomare Domänen und die erste Normalform (1NF)
- 4.9.3. Dekomposition durch funktionale Abhängigkeiten
- 4.9.4. Funktionale Abhängigkeitstheorie
- 4.9.5. Zersetzungsalgorithmen
- 4.9.6. Zerlegung unter Verwendung mehrwertiger Abhängigkeiten
- 4.9.7. Weitere Normalformen
- 4.9.8. Prozess der Datenbankentwicklung

4.10. NoSQL-Datenbanken
- 4.10.1. Was sind NoSQL-Datenbanken?
- 4.10.2. Analyse der verschiedenen NoSQL-Optionen und ihrer Eigenschaften
- 4.10.3. MongoDB

Modul 5. Fortgeschrittene Datenbanken

5.1. Einführung in verschiedene Datenbanksysteme
- 5.1.1. Historischer Rückblick
- 5.1.2. Hierarchisch-Datenbanken
- 5.1.3. Netzwerk-Datenbanken
- 5.1.4. Relationale Datenbanken
- 5.1.5. Nicht-relationale Datenbanken

5.2. XML und Datenbanken für das Web
- 5.2.1. Validierung von XML-Dokumenten
- 5.2.2. Transformationen von XML-Dokumenten
- 5.2.3. XML-Datenspeicherung
- 5.2.4. XML-relationale Datenbanken
- 5.2.5. SQL/XML
- 5.2.6. Native XML-Datenbanken

5.3. Parallele Datenbanken
- 5.3.1. Parallele Systeme
- 5.3.2. Parallele Datenbankarchitekturen
- 5.3.4. Parallelität bei Konsultationen
- 5.3.5. Parallelität zwischen den Konsultationen
- 5.3.6. Paralleler Systementwurf
- 5.3.7. Parallelverarbeitung in SQL

5.4. Verteilte Datenbanken
- 5.4.1. Verteilte Systeme
- 5.4.2. Verteilte Speicherung
- 5.4.3. Verfügbarkeit
- 5.4.4. Verteilte Abfrageverarbeitung
- 5.4.5. Anbieter verteilter Datenbanken

5.5. Indexierung und Assoziierung
- 5.5.1. Geordnete Indizes
- 5.5.2. Dichte und spärliche Indizes
- 5.5.3. Multilevel-Indizes
- 5.5.4. Aktualisierung des Indexes
- 5.5.5. Statische Assoziation
- 5.5.6. Wie man Indizes in Datenbanken verwendet

5.6. Einführung in die transaktionale Verarbeitung
- 5.6.1. Zustände einer Transaktion
- 5.6.2. Implementierung von Atomarität und Dauerhaftigkeit
- 5.6.3. Sequenzierung
- 5.6.4. Wiederherstellbarkeit
- 5.6.5. Implementierung der Isolierung

5.7. Rückgewinnungssysteme
- 5.7.1. Klassifizierung von Fehlern
- 5.7.2. Strukturen zur Lagerung
- 5.7.3. Rückforderung und Atomarität
- 5.7.4. Rückforderung auf der Grundlage historischer Daten
- 5.7.5. Gleichzeitige Transaktionen und Wiederherstellung
- 5.7.6. Hohe Verfügbarkeit von Datenbanken

5.8. Ausführung und Bearbeitung von Abfragen
- 5.8.1. Kosten für eine Beratung
- 5.8.2. Auswahlverfahren
- 5.8.3. Ordination
- 5.8.4. Einführung in die Abfrageoptimierung
- 5.8.5. Leistungsüberwachung

5.9. Nicht-relationale Datenbanken
- 5.9.1. Dokumentorientierte Datenbanken
- 5.9.2. Graphorientierte Datenbanken
- 5.9.3. Key-Value-Datenbanken

5.10. Data Warehouse, OLAP und Data Mining
- 5.10.1. Komponenten von Data Warehouses
- 5.10.2. Architektur eines Data Warehouse
- 5.10.3. OLAP
- 5.10.4. Data-Mining-Funktionen
- 5.10.5. Andere Arten von Mining

Modul 6. Fortgeschrittener Algorithmusentwurf

6.1. Analyse von Rekursions-und Divide-and-Conquer-Algorithmen
- 6.1.1. Aufstellen und Lösen von homogenen und nicht-homogenen Rekursionsgleichungen
- 6.1.2. Überblick über die Strategie des Divide-and-Conquer

6.2. Amortisierte Analyse
- 6.2.1. Aggregierte Analyse
- 6.2.2. Die Rechnungslegungsmethode
- 6.2.3. Die Potenzialmethode

6.3. Dynamische Programmierung und Algorithmen für NP-Probleme
- 6.3.1. Merkmale der dynamischen Programmierung
- 6.3.2. Zurückverfolgung: *Backtracking*
- 6.3.3. Verzweigung und Beschneidung

6.4. Kombinatorische Optimierung
- 6.4.1. Darstellung der Probleme
- 6.4.2. 1D-Optimierung

6.5. Algorithmen zur Randomisierung
- 6.5.1. Beispiele für Randomisierungsalgorithmen
- 6.5.2. Das Buffonsche Theorem
- 6.5.3. Monte-Carlo- Algorithmus
- 6.5.4. Las-Vegas-Algorithmus

6.6. Lokale Suche und Kandidatensuche
- 6.6.1. *Gradient Ascent*
- 6.6.2. *Hill Climbing*
- 6.6.3. *Simulated Annealing*
- 6.6.4. *Tabu Search*
- 6.6.5. Suche mit Kandidaten

6.7. Formale Überprüfung von Programmen
- 6.7.1. Spezifikation von funktionalen Abstraktionen
- 6.7.2. Die Sprache der Logik erster Ordnung
- 6.7.3. Das formale System von Hoare

6.8. Überprüfung der iterativen Programme
- 6.8.1. Regeln des formalen Hoare-Systems
- 6.8.2. Konzept der invarianten Iterationen

6.9. Numerische Methoden
- 6.9.1. Die Methode der Halbierung
- 6.9.2. Newton-Raphson-Methode
- 6.9.3. Die Sekantenmethode

6.10. Parallele Algorithmen
- 6.10.1. Parallele Binäroperationen
- 6.10.2. Parallele Operationen mit Graphs
- 6.10.3. Parallelität im Devide-and-Conquer
- 6.10.4. Parallelität in der dynamischen Programmierung

Modul 7. Mensch-Computer Interaktion

7.1. Einführung in die Mensch-Computer Interaktion
- 7.1.1. Was ist die Mensch-Computer Interaktion?
- 7.1.2. Beziehung der Mensch-Computer Interaktion zu anderen Disziplinen
- 7.1.3. Die Benutzeroberfläche
- 7.1.4. Benutzerfreundlichkeit und Zugänglichkeit
- 7.1.5. Benutzererfahrung und benutzerzentriertes Design

7.2. Der Computer und die Interaktion: Benutzeroberfläche und Interaktionsparadigmen
- 7.2.1. Interaktion
- 7.2.2. Interaktionsparadigmen und Interaktionsstile
- 7.2.3. Entwicklung von Benutzeroberflächen
- 7.2.4. Klassische Benutzeroberflächen: WIMP/GUI, Befehle, Stimme, Virtuelle Realität
- 7.2.5. Innovative Benutzeroberflächen: mobil, tragbar, kollaborativ, BCI

7.3. Der menschliche Faktor: psychologische und kognitive Aspekte
- 7.3.1. Die Bedeutung des menschlichen Faktors in der Interaktion
- 7.3.2. Menschliche Informationsverarbeitung
- 7.3.3. Die Eingabe und Ausgabe von Informationen: visuell, auditiv und taktil
- 7.3.4. Wahrnehmung und Aufmerksamkeit
- 7.3.5. Wissen und mentale Modelle: Darstellung, Organisation und Erwerb

7.4. Der menschliche Faktor: sensorische und physische Einschränkungen
- 7.4.1. Funktionelle Vielfalt, Behinderung und Beeinträchtigung
- 7.4.2. Visuelle Vielfalt
- 7.4.3. Akustische Vielfalt
- 7.4.4. Kognitive Vielfalt
- 7.4.5. Motorische Vielfalt
- 7.4.6. Der Fall der digitalen Einwanderer

7.5. Der Designprozess (I): Anforderungsanalyse für die Gestaltung der Benutzeroberfläche
- 7.5.1. Benutzerzentriertes Design
- 7.5.2. Was ist eine Anforderungsanalyse?
- 7.5.3. Sammeln von Informationen
- 7.5.4. Analyse und Interpretation der Informationen
- 7.5.5. Der Designprozess (II): Prototyping und Aufgabenanalyse

7.6. Analyse der Benutzerfreundlichkeit und Zugänglichkeit
- 7.6.1. Konzeptioneller Entwurf
- 7.6.2. Prototyping
- 7.6.3. Hierarchische Aufgabenanalyse

7.7. Der Designprozess (III): Bewertung
- 7.7.1. Bewertung im Designprozess: Ziele und Methoden
- 7.7.2. Bewertungsmethoden ohne Benutzer
- 7.7.3. Bewertungsmethoden mit Benutzern
- 7.7.4. Bewertungsstandards und -normen

7.8. Zugänglichkeit: Definition und Leitlinien
- 7.8.1. Barrierefreiheit und universelles Design
- 7.8.2. WAI-Initiative und WCAG-Richtlinien
- 7.8.3. WCAG-Richtlinien 2.0 und 2.1

7.9. Zugänglichkeit: Bewertung und Funktionsvielfalt
- 7.9.1. Tools zur Bewertung der Barrierefreiheit im Web
- 7.9.2. Zugänglichkeit und Funktionsvielfalt

7.10. Der Computer und die Interaktion: Peripheriegeräte und Zubehör
- 7.10.1. Herkömmliche und Peripheriegeräte
- 7.10.2. Alternative und Peripheriegeräte
- 7.10.3. Handys und Tablets
- 7.10.4. Funktionsvielfalt, Interaktion und Peripheriegeräte

Modul 8. Fortgeschrittene Programmierung

8.1. Einführung in die objektorientierte Programmierung
- 8.1.1. Einführung in die objektorientierte Programmierung
- 8.1.2. Klassen-Design
- 8.1.3. Einführung in UML für die Modellierung von Problemen

8.2. Beziehungen zwischen den Klassen
- 8.2.1. Abstraktion und Vererbung
- 8.2.2. Fortgeschrittene Konzepte der Vererbung
- 8.2.3. Polymorphismen
- 8.2.4. Zusammensetzung und Aggregation

8.3. Einführung in Design Patterns für objektorientierte Probleme
- 8.3.1. Was sind Entwurfsmuster
- 8.3.2. *Factory* Muster
- 8.3.4. *Singleton* Muster
- 8.3.5. *Observer* Muster
- 8.3.6. *Composite* Muster

8.4. Ausnahmen
- 8.4.1. Welche Ausnahmen gibt es?
- 8.4.2. Abfangen und Behandlung von Ausnahmen
- 8.4.3. Starten von Ausnahmen
- 8.4.4. Erstellung von Ausnahmen

8.5. Benutzeroberflächen
- 8.5.1. Einführung in Qt
- 8.5.2. Positionierung
- 8.5.3. Was sind Ereignisse?
- 8.5.4. Ereignisse: Definition und Erfassung
- 8.5.5. Entwicklung von Benutzeroberflächen

8.6. Einführung in die gleichzeitige Programmierung
- 8.6.1. Einführung in die gleichzeitige Programmierung
- 8.6.2. Das Konzept des Prozesses und des Threads
- 8.6.3. Interaktion zwischen Prozessen oder Threads
- 8.6.4. Threads im C++
- 8.6.6. Vor- und Nachteile der gleichzeitigen Programmierung

8.7. Thread-Management und Synchronisierung
- 8.7.1. Lebenszyklus von Threads
- 8.7.2. Die *Thread* Klasse
- 8.7.3. Planung von Threads
- 8.7.4. Threads-Gruppen
- 8.7.5. Dämonenartige Threads
- 8.7.6. Synchronisierung
- 8.7.7. Verriegelungsmechanismen
- 8.7.8. Kommunikationsmechanismen
- 8.7.9. Monitore

8.8. Häufige Probleme bei der gleichzeitigen Programmierung
- 8.8.1. Das Problem der Erzeuger/Verbraucher
- 8.8.2. Das Problem der Leser und der Schreiber
- 8.8.3. Das Problem des Abendessens der Philosophen

8.9. Dokumentation und Prüfung von Software
- 8.9.1. Warum ist es wichtig, Software zu dokumentieren?
- 8.9.2. Design-Dokumentation
- 8.9.3. Einsatz von Tools zur Dokumentation

8.10. Software-Tests
- 8.10.1. Einführung in die Softwaretests
- 8.10.2. Arten von Tests
- 8.10.3. Einheitstest
- 8.10.4. Integrationstests
- 8.10.5. Validierungstest
- 8.10.6. Systemtest

Modul 9. Entwicklung vernetzter Anwendungen

9.1. HTML5-Auszeichnungssprachen
- 9.1.1. Grundlegende Konzepte der HTML
- 9.1.2. Neue HTML 5-Elemente
- 9.1.3. Formulare: neue Steuerelemente

9.2. Einführung in CSS-Stylesheets
- 9.2.1. Erste Schritte mit CSS
- 9.2.2. Einführung in CSS3

9.3. Browser-Skriptsprache: JavaScript
- 9.3.1. JavaScript-Grundlagen
- 9.3.2. DOM
- 9.3.3. Ereignisse
- 9.3.4. jQuery
- 9.3.5. Ajax

9.4. Konzept der komponentenorientierten Programmierung
- 9.4.1. Kontext
- 9.4.2. Komponenten und Schnittstellen
- 9.4.3. Zustände einer Komponente

9.5. Architektur von Komponenten
- 9.5.1. Zeitgenössische Architekturen
- 9.5.2. Integration und Einsatz von Komponenten

9.6. *Framework Frontend*: Bootstrap
- 9.6.1. Rasterdesign
- 9.6.2. Formulare
- 9.6.3. Komponenten

9.7. Model-View-Controller
- 9.7.1. Methoden der Webentwicklung
- 9.7.2. Entwurfsmuster: MVC

9.8. Grid-Informationstechnologien
- 9.8.1. Erhöhte Computerressourcen
- 9.8.2. Konzept der Grid-Technologie

9.9. Serviceorientierte Architektur
- 9.9.1. SOA und Webdienste
- 9.9.2. Topologie eines Webdienstes
- 9.9.3. Plattformen für Webdienste

9.10. HTTP-Protokoll
- 9.10.1. Nachrichten
- 9.10.2. Dauerhafte Sitzungen
- 9.10.3. Kryptographisches System
- 9.10.4. Funktionsweise des HTTPS-Protokolls

Modul 10. Softwareentwicklung

10.1. Einführung in die Softwareentwicklung und -modellierung
- 10.1.1. Die Natur der Software
- 10.1.2. Die einzigartige Natur von WebApps
- 10.1.3. Softwareentwicklung
- 10.1.4. Der Software-Prozess
- 10.1.5. Die Praxis der Softwareentwicklung
- 10.1.6. Software-Mythen
- 10.1.7. Wie alles beginnt
- 10.1.8. Objektorientierte Konzepte
- 10.1.9. Einführung in UML

10.2. Der Software-Prozess
- 10.2.1. Ein allgemeines Prozessmodell
- 10.2.2. Vorgeschriebene Prozessmodelle
- 10.2.3. Spezialisierte Prozessmodelle
- 10.2.4. Der vereinheitlichte Prozess
- 10.2.5. Personal- und Teamprozessmodelle
- 10.2.6. Was ist Agilität?
- 10.2.7. Was ist ein agiler Prozess?
- 10.2.8. Scrum
- 10.2.9. Werkzeugkasten für agile Prozesse

10.3. Grundsätze für die Praxis der Softwareentwicklung
- 10.3.1. Leitprinzipien des Prozesses
- 10.3.2. Prinzipien als Leitfaden für die Praxis
- 10.3.3. Grundsätze der Kommunikation
- 10.3.4. Grundsätze der Planung
- 10.3.5. Grundsätze der Modellierung
- 10.3.6. Konstruktionsprinzipien
- 10.3.7. Grundsätze für die Einführung

10.4. Verständnis der Anforderungen
- 10.4.1. Anforderungsmanagement
- 10.4.2. Schaffung der Grundlagen
- 10.4.3. Bedarfsermittlung
- 10.4.4. Entwicklung von Anwendungsfällen
- 10.4.5. Ausarbeitung des Anforderungsmodells
- 10.4.6. Aushandeln von Anforderungen
- 10.4.7. Validierung der Anforderungen

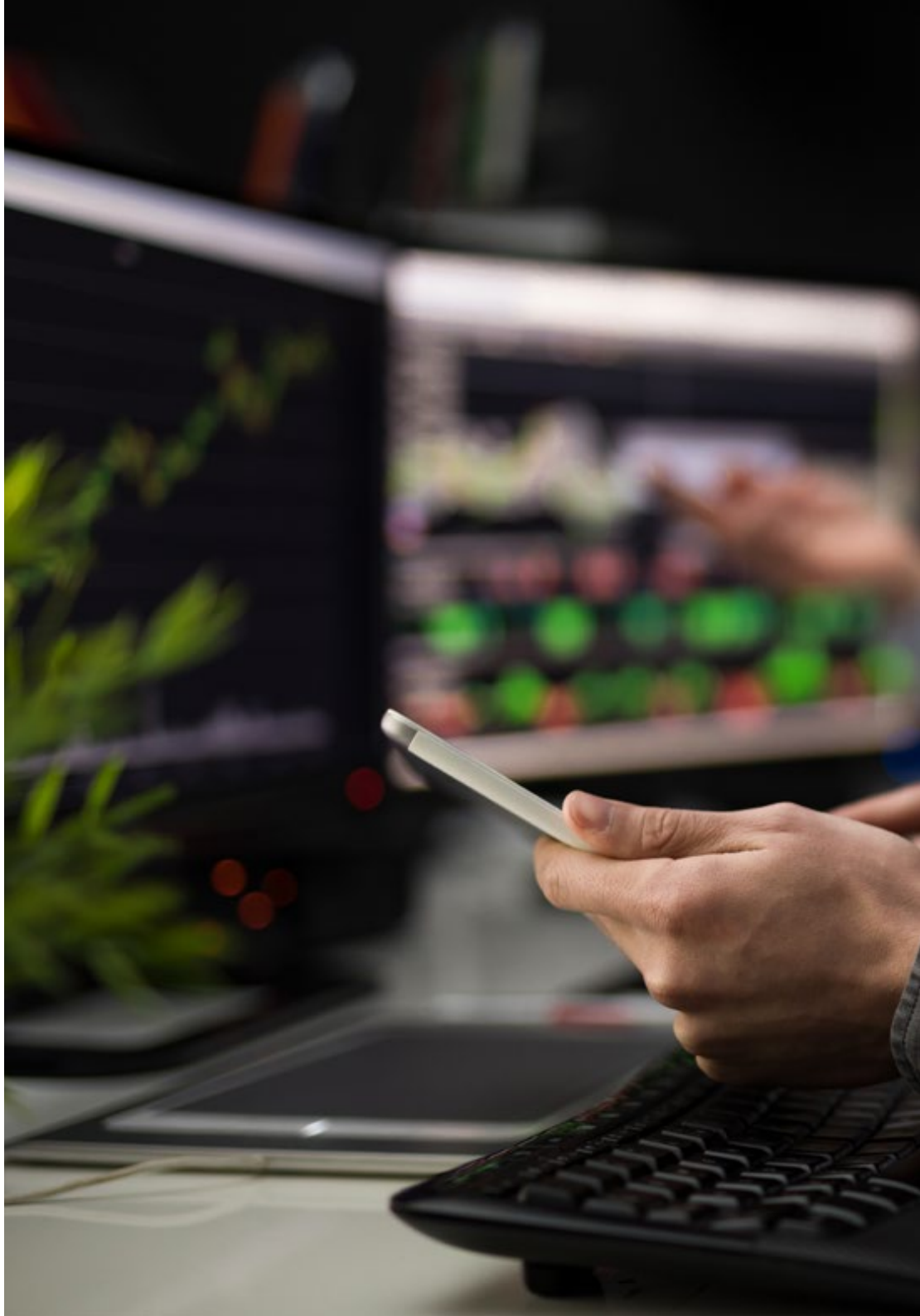
10.5. Anforderungsmodellierung: Szenarien, Informations- und Analyseklassen
- 10.5.1. Analyse der Anforderungen
- 10.5.2. Szenario-basiertes Modell
- 10.5.3. UML-Modelle, die den Anwendungsfall liefern
- 10.5.4. Konzepte der Datenmodellierung
- 10.5.5. Klassen-basiertes Modell
- 10.5.6. Klassendiagramme

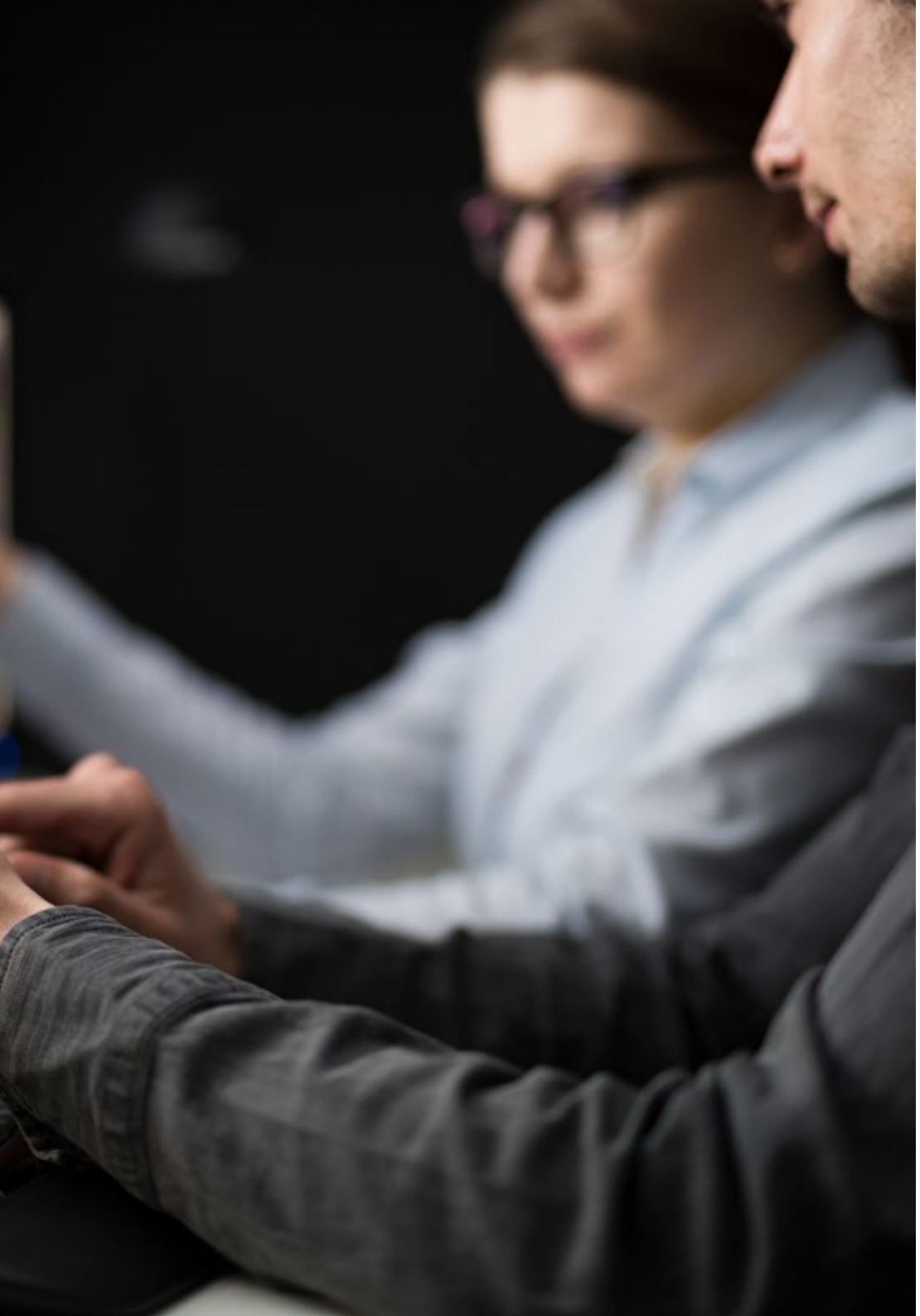
10.6. Anforderungsmodellierung: Fluss, Verhalten und Muster
- 10.6.1. Anforderungen die die Strategien gestalten
- 10.6.2. Flussorientierte Modellierung
- 10.6.3. Zustandsdiagramme
- 10.6.4. Erstellung eines Verhaltensmodells
- 10.6.5. Sequenzdiagramme
- 10.6.6. Kommunikationsdiagramme
- 10.6.7. Muster für die Modellierung von Anforderungen

10.7. Konzepte des Designs
- 10.7.1. Design im Kontext der Softwareentwicklung
- 10.7.2. Der Entwurfsprozess
- 10.7.3. Konzepte des Designs
- 10.7.4. Objektorientierte Konzepte des Designs
- 10.7.5. Das Designmodell

10.8. Design der Architektur
- 10.8.1. Software-Architektur
- 10.8.2. Architektonische Gattungen
- 10.8.3. Architektonische Stile
- 10.8.4. Architektonischer Design
- 10.8.5. Entwicklung von alternativen Designs für die Architektur
- 10.8.6. Abbildung der Architektur mit Hilfe von Datenflüssen

10.9. Design auf Komponentenebene und musterbasierter Entwurf
- 10.9.1. Was ist eine Komponente?
- 10.9.2. Klassenbasiertes Komponentendesign
- 10.9.3. Verwirklichung des Designs auf Komponentenebene
- 10.9.4. Design der traditionellen Komponenten
- 10.9.5. Komponentenbasierte Entwicklung
- 10.9.6. Entwurfsmuster
- 10.9.7. Musterbasiertea Softwaredesign
- 10.9.8. Architektonische Muster
- 10.9.9. Musterdesign auf Komponentenebene
- 10.9.10. Musterdesign für Benutzeroberflächen





10.10. Softwarequalität und Projektmanagement
- 10.10.1. Qualität
- 10.10.1. Software Qualität
- 10.10.2. Das Dilemma der Softwarequalität
- 10.10.3. Erreichen von Softwarequalität
- 10.10.4. Software-Qualitätssicherung
- 10.10.5. Das administrative Spektrum
- 10.10.6. Personal
- 10.10.7. Das Produkt
- 10.10.8. Der Prozess
- 10.10.9. Das Projekt
- 10.10.10. Grundsätze und Praktiken

Eine einzigartige, wichtige und entscheidende Fortbildungserfahrung, die Ihre berufliche Entwicklung fördert"

05

Methodik

Dieses Fortbildungsprogramm bietet eine andere Art des Lernens. Unsere Methodik wird durch eine zyklische Lernmethode entwickelt: ***das Relearning***.

Dieses Lehrsystem wird z. B. an den renommiertesten medizinischen Fakultäten der Welt angewandt und wird von wichtigen Publikationen wie dem ***New England Journal of Medicine*** als eines der effektivsten angesehen.

Entdecken Sie Relearning, ein System, das das herkömmliche lineare Lernen aufgibt und Sie durch zyklische Lehrsysteme führt: eine Art des Lernens, die sich als äußerst effektiv erwiesen hat, insbesondere in Fächern, die Auswendiglernen erfordern"

Fallstudie zur Kontextualisierung aller Inhalte

Unser Programm bietet eine revolutionäre Methode zur Entwicklung von Fähigkeiten und Kenntnissen. Unser Ziel ist es, Kompetenzen in einem sich wandelnden, wettbewerbsorientierten und sehr anspruchsvollen Umfeld zu stärken.

Mit TECH werden Sie eine Art des Lernens erleben, die die Grundlagen der traditionellen Universitäten in der ganzen Welt verschiebt"

Sie werden Zugang zu einem Lernsystem haben, das auf Wiederholung basiert, mit natürlichem und progressivem Unterricht während des gesamten Lehrplans.

Die Studenten lernen durch gemeinschaftliche Aktivitäten und reale Fälle die Lösung komplexer Situationen in realen Geschäftsumgebungen.

Eine innovative und andersartige Lernmethode

Dieses TECH-Programm ist ein von Grund auf neu entwickeltes, intensives Lehrprogramm, das die anspruchsvollsten Herausforderungen und Entscheidungen in diesem Bereich sowohl auf nationaler als auch auf internationaler Ebene vorsieht. Dank dieser Methodik wird das persönliche und berufliche Wachstum gefördert und ein entscheidender Schritt in Richtung Erfolg gemacht. Die Fallmethode, die Technik, die diesem Inhalt zugrunde liegt, gewährleistet, dass die aktuellste wirtschaftliche, soziale und berufliche Realität berücksichtigt wird.

> *Unser Programm bereitet Sie darauf vor, sich neuen Herausforderungen in einem unsicheren Umfeld zu stellen und in Ihrer Karriere erfolgreich zu sein"*

Die Fallmethode ist das am weitesten verbreitete Lernsystem an den besten Informatikschulen der Welt, seit es sie gibt. Die Fallmethode wurde 1912 entwickelt, damit die Jurastudenten das Recht nicht nur anhand theoretischer Inhalte erlernen, sondern ihnen reale, komplexe Situationen vorlegen, damit sie fundierte Entscheidungen treffen und Werturteile darüber fällen können, wie diese zu lösen sind. Sie wurde 1924 als Standardlehrmethode in Harvard eingeführt.

Was sollte eine Fachkraft in einer bestimmten Situation tun? Mit dieser Frage konfrontieren wir Sie in der Fallmethode, einer handlungsorientierten Lernmethode. Während des gesamten Kurses werden die Studierenden mit mehreren realen Fällen konfrontiert. Sie müssen Ihr gesamtes Wissen integrieren, recherchieren, argumentieren und Ihre Ideen und Entscheidungen verteidigen.

Relearning Methodik

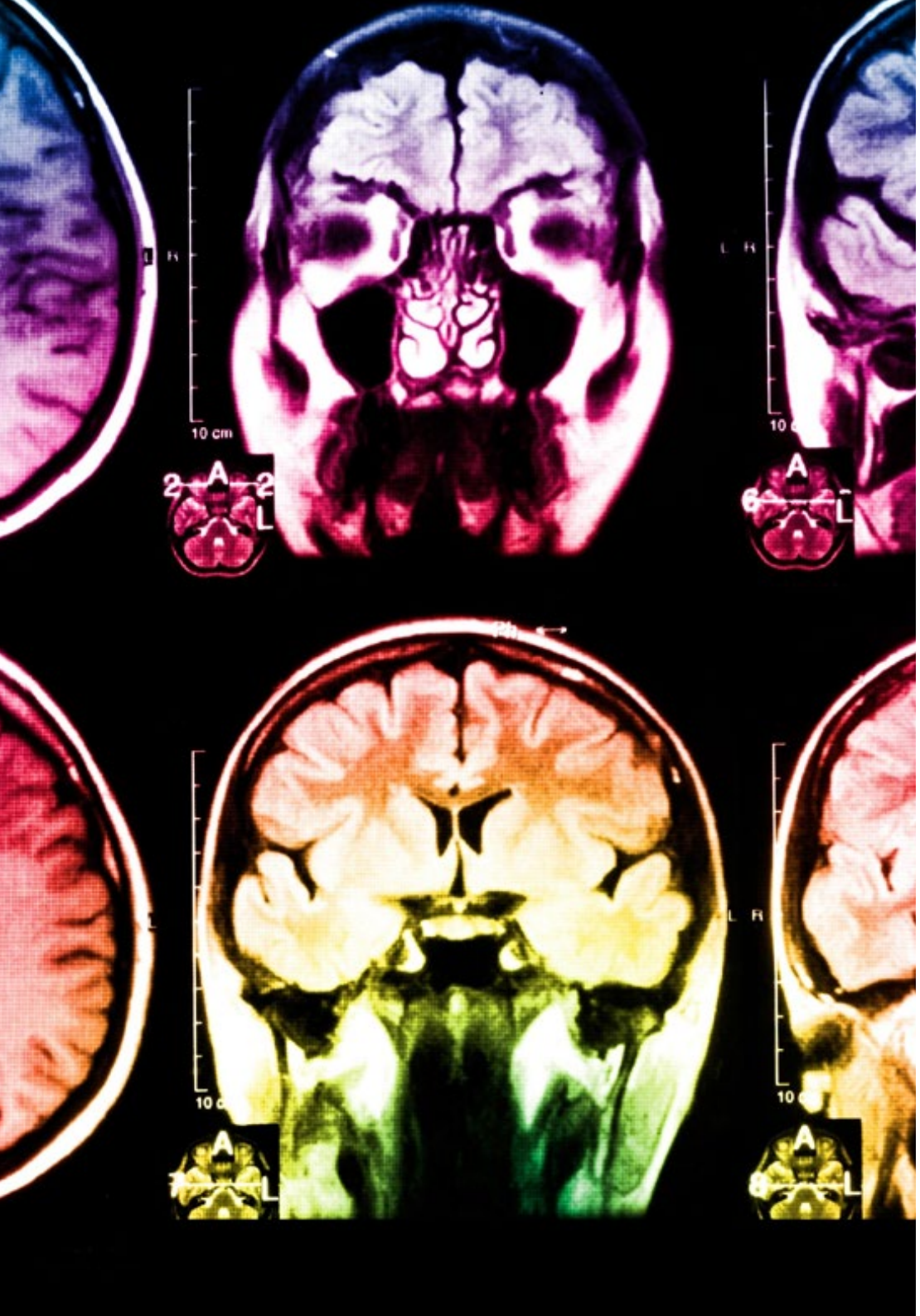
TECH kombiniert die Methodik der Fallstudien effektiv mit einem 100%igen Online-Lernsystem, das auf Wiederholung basiert und in jeder Lektion verschiedene didaktische Elemente kombiniert.

Wir ergänzen die Fallstudie mit der besten 100%igen Online-Lehrmethode: Relearning.

Im Jahr 2019 erzielten wir die besten Lernergebnisse aller spanischsprachigen Online-Universitäten der Welt.

Bei TECH lernen Sie mit einer hochmodernen Methodik, die darauf ausgerichtet ist, die Führungskräfte der Zukunft auszubilden. Diese Methode, die an der Spitze der weltweiten Pädagogik steht, wird Relearning genannt.

Unsere Universität ist die einzige in der spanischsprachigen Welt, die für die Anwendung dieser erfolgreichen Methode zugelassen ist. Im Jahr 2019 ist es uns gelungen, die Gesamtzufriedenheit unserer Studenten (Qualität der Lehre, Qualität der Materialien, Kursstruktur, Ziele...) in Bezug auf die Indikatoren der besten Online-Universität in Spanisch zu verbessern.



In unserem Programm ist das Lernen kein linearer Prozess, sondern erfolgt in einer Spirale (lernen, verlernen, vergessen und neu lernen). Daher wird jedes dieser Elemente konzentrisch kombiniert. Mit dieser Methode wurden mehr als 650.000 Hochschulabsolventen mit beispiellosem Erfolg in so unterschiedlichen Bereichen wie Biochemie, Genetik, Chirurgie, internationales Recht, Managementfähigkeiten, Sportwissenschaft, Philosophie, Recht, Ingenieurwesen, Journalismus, Geschichte, Finanzmärkte und -Instrumente ausgebildet. Dies alles in einem sehr anspruchsvollen Umfeld mit einer Studentenschaft mit hohem sozioökonomischem Profil und einem Durchschnittsalter von 43,5 Jahren.

Das Relearning ermöglicht es Ihnen, mit weniger Aufwand und mehr Leistung zu lernen, sich mehr auf Ihr Fachgebiet einzulassen, einen kritischen Geist zu entwickeln, Argumente zu verteidigen und Meinungen zu kontrastieren: eine direkte Gleichung zum Erfolg.

Nach den neuesten wissenschaftlichen Erkenntnissen der Neurowissenschaften wissen wir nicht nur, wie wir Informationen, Ideen, Bilder und Erinnerungen organisieren, sondern auch, dass der Ort und der Kontext, in dem wir etwas gelernt haben, von grundlegender Bedeutung dafür sind, dass wir uns daran erinnern und es im Hippocampus speichern können, um es in unserem Langzeitgedächtnis zu behalten.

Auf diese Weise sind die verschiedenen Elemente unseres Programms im Rahmen des so genannten neurokognitiven kontextabhängigen E-Learnings mit dem Kontext verbunden, in dem der Teilnehmer seine berufliche Praxis entwickelt.

Dieses Programm bietet die besten Lehrmaterialien, die sorgfältig für Fachleute aufbereitet sind:

Studienmaterial

Alle didaktischen Inhalte werden von den Fachleuten, die den Kurs unterrichten werden, speziell für den Kurs erstellt, so dass die didaktische Entwicklung wirklich spezifisch und konkret ist.

Diese Inhalte werden dann auf das audiovisuelle Format angewendet, um die TECH-Online-Arbeitsmethode zu schaffen. Und das alles mit den neuesten Techniken, die dem Studenten qualitativ hochwertige Stücke aus jedem einzelnen Material zur Verfügung stellen.

Meisterklassen

Die Nützlichkeit der Expertenbeobachtung ist wissenschaftlich belegt.

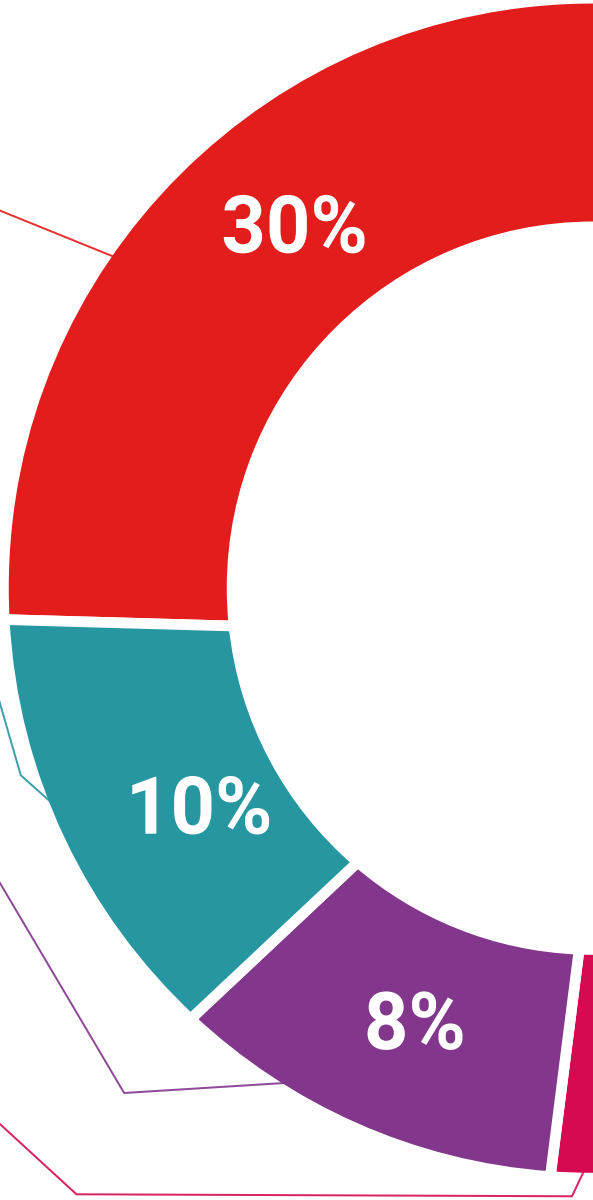
Das sogenannte Learning from an Expert baut Wissen und Gedächtnis auf und schafft Vertrauen für zukünftige schwierige Entscheidungen.

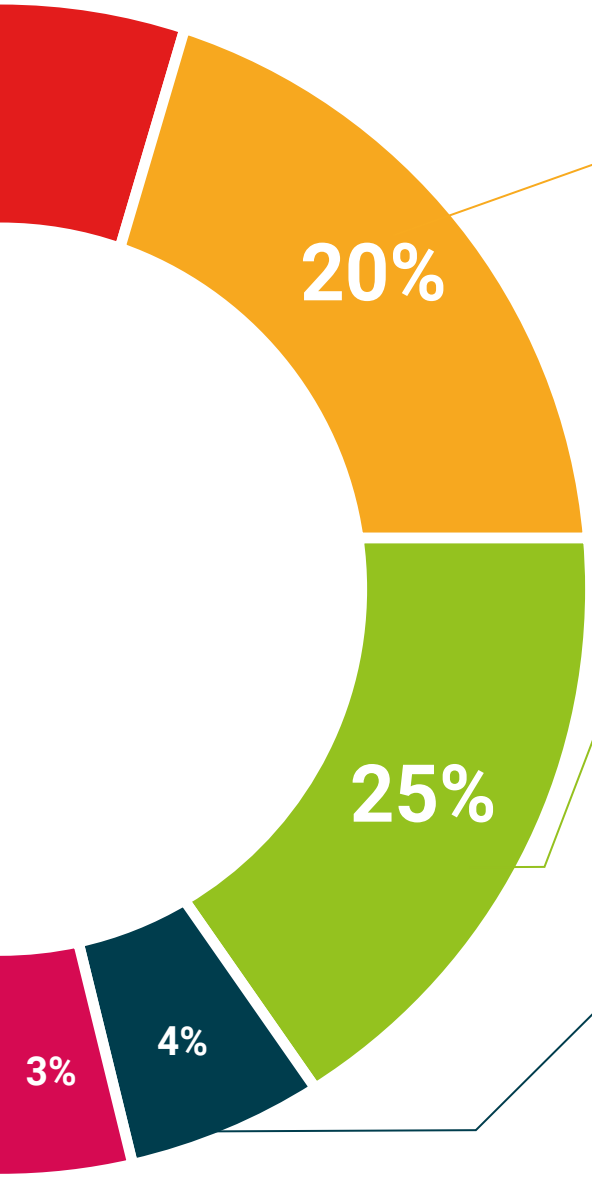
Fertigkeiten und Kompetenzen Praktiken

Sie werden Aktivitäten durchführen, um spezifische Kompetenzen und Fertigkeiten in jedem Fachbereich zu entwickeln. Praktiken und Dynamiken zum Erwerb und zur Entwicklung der Fähigkeiten und Fertigkeiten, die ein Spezialist im Rahmen der Globalisierung, in der wir leben, entwickeln muss.

Weitere Lektüren

Aktuelle Artikel, Konsensdokumente und internationale Leitfäden, u.a. In der virtuellen Bibliothek von TECH haben die Studenten Zugang zu allem, was sie für ihre Ausbildung benötigen.





Fallstudien

Sie werden eine Auswahl der besten Fallstudien vervollständigen, die speziell für diese Qualifizierung ausgewählt wurden. Die Fälle werden von den besten Spezialisten der internationalen Szene präsentiert, analysiert und betreut.

Interaktive Zusammenfassungen

Das TECH-Team präsentiert die Inhalte auf attraktive und dynamische Weise in multimedialen Pillen, die Audios, Videos, Bilder, Diagramme und konzeptionelle Karten enthalten, um das Wissen zu vertiefen.

Dieses einzigartige Bildungssystem für die Präsentation multimedialer Inhalte wurde von Microsoft als "europäische Erfolgsgeschichte" ausgezeichnet.

Prüfung und Nachprüfung

Die Kenntnisse der Studenten werden während des gesamten Programms regelmäßig durch Bewertungs- und Selbsteinschätzungsaktivitäten und -übungen beurteilt und neu bewertet, so dass die Studenten überprüfen können, wie sie ihre Ziele erreichen.

06

Qualifizierung

Der Privater Masterstudiengang in Softwareentwicklung garantiert neben der strengsten und aktuellsten Ausbildung auch den Zugang zu einem von der TECH Technologischen Universität ausgestellten Diplom.

Schließen Sie dieses Programm erfolgreich ab und erhalten Sie Ihren Universitätsabschluss ohne lästige Reisen oder Formalitäten"

Dieser **Privater Masterstudiengang in Softwareentwicklung** enthält das vollständigste und aktuellste Programm auf dem Markt.

Sobald der Student die Prüfungen bestanden hat, erhält er/sie per Post* mit Empfangsbestätigung das entsprechende Diplom, ausgestellt von der **TECH Technologischen Universität**.

Das von **TECH Technologische Universität** ausgestellte Diplom drückt die erworbene Qualifikation aus und entspricht den Anforderungen, die in der Regel von Stellenbörsen, Auswahlprüfungen und Berufsbildungsausschüssen verlangt werden.

Titel: **Privater Masterstudiengang in Softwareentwicklung**

Anzahl der offiziellen Arbeitsstunden: **1.500 Std.**


tech technologische universität

Verleiht dieses

DIPLOM

an

Herr/Frau _____, mit Ausweis-Nr. _____

Für den erfolgreichen Abschluss und die Akkreditierung des Programms

PRIVATER MASTERSTUDIENGANG

in

Softwareentwicklung

Es handelt sich um einen von dieser Universität verliehenen Abschluss, mit einer Dauer von 1.500 Stunden, mit Anfangsdatum tt/mm/jjjj und Enddatum tt/mm/jjjj.

TECH ist eine private Hochschuleinrichtung, die seit dem 28. Juni 2018 vom Ministerium für öffentliche Bildung anerkannt ist.

Zum 17. Juni 2020

Tere Guevara Navarro
Rectora

Diese Qualifikation muss immer mit einem Hochschulabschluss einhergehen, der von der für die Berufsausübung zuständigen Behörde des jeweiligen Landes ausgestellt wurde. einzigartiger Code TECH: AFWOR23S techtitute.com/titel


Privater Masterstudiengang in Softwareentwicklung

Fachkategorie	Stunden
Obligatorisch (OB)	1.500
Wahlfach(OP)	0
Externes Praktikum (PR)	0
Masterarbeit (TFM)	0
	Summe 1.500

Allgemeiner Aufbau des Lehrplans

Kurs	Modul	Stunden	Kategorie
1°	Grundlagen der Programmierung	150	OB
1°	Datenstruktur	150	OB
1°	Algorithmus und Komplexität	150	OB
1°	Datenbanken	150	OB
1°	Fortgeschrittene Datenbanken	150	OB
1°	Fortgeschrittener Algorithmusentwurf	150	OB
1°	Mensch-Computer Interaktion	150	OB
1°	Fortgeschrittene Programmierung	150	OB
1°	Entwicklung vernetzter Anwendungen	150	OB
1°	Softwareentwicklung	150	OB

Tere Guevara Navarro
Rectora

tech technologische universität

*Haager Apostille. Für den Fall, dass der Student die Haager Apostille für sein Papierdiplom beantragt, wird TECH EDUCATION die notwendigen Vorkehrungen treffen, um diese gegen eine zusätzliche Gebühr zu beschaffen.

tech technologische universität

Privater Masterstudiengang

Softwareentwicklung

- Modalität: **online**
- Dauer: **12 Monate**
- Qualifizierung: **TECH Technologische Universität**
- Aufwand: **16 Std./Woche**
- Zeitplan: **in Ihrem eigenen Tempo**
- Prüfungen: **online**

Privater Masterstudiengang
Softwareentwicklung

tech technologische universität