Master Privato
Sviluppo di Software

tech università tecnologica

tech università tecnologica

Master Privato
Sviluppo di Software

- Modalità: **online**
- Durata: **12 mesi**
- Titolo: **TECH Università Tecnologica**
- Dedizione: **16 ore/settimana**
- Orario: **a scelta**
- Esami: **online**

Accesso al sito web: www.techtitute.com/it/informatica/master/master-sviluppo-software

Indice

01

Presentazione

Per partecipare a una delle aree di maggiore proiezione del settore informatico, il professionista deve essere preparato scientificamente e tecnologicamente ad affrontare in modo efficace le sfide che si presentano nella pratica professionale dell'ingegneria del software. Questo Master Privato si propone perciò di fornire un elevato livello di padronanza nello Sviluppo del Software, grazie ai più recenti progressi e sviluppi in questo campo e per mezzo di una metodologia di studio di massimo impatto e di straordinaria flessibilità.

Acquisisci le conoscenze più complete nell'ambito dell'ingegneria del software ottenendo la qualifica più aggiornata del mercato educativo online e inizia a lavorare sugli sviluppi di questo dinamico settore professionale"

Il software è diventato un elemento estremamente importante nel mondo di oggi a seguito dei progressi delle nuove tecnologie. Negli ultimi anni è emersa l'esigenza di poter sviluppare prodotti software che abbiano la giusta funzionalità e qualità, rispettando i tempi e il budget.

Questo programma è rivolto alle persone interessate a raggiungere un livello di conoscenza superiore nello Sviluppo di Software. L'obiettivo principale è quello di consentire agli studenti di applicare le conoscenze acquisite in questo Master Privato nel mondo reale, in un ambiente di lavoro che riproduce le condizioni che potrebbero incontrare in futuro, in modo rigoroso e realistico.

Cogli l'opportunità di studiare questa qualifica in modalità 100% online, senza dover rinunciare ai tuoi impegni e tornando in modo più facile all'università. Aggiorna le tue conoscenze e ottieni un titolo di Master Privato per continuare a crescere personalmente e professionalmente.

Acquisirai conoscenze approfondite nel campo dell'ingegneria del software, ma anche nel campo dell'informatica e della struttura dei computer, comprese le basi matematiche, statistiche e fisiche essenziali per l'ingegneria.

Cogli l'opportunità e formati in maniera 100% online, senza dover rinunciare ai tuoi impegni e rendendo facile il tuo ritorno all'università. Aggiorna le tue conoscenze e ottieni il tuo titolo di master per continuare a crescere personalmente e professionalmente.

Questo **Master Privato in Sviluppo di Software** possiede il programma più completo e aggiornato del mercato. Le caratteristiche principali del programma sono:

- Studio di 100 scenari simulati presentati da esperti in Sviluppo di Software
- Contenuti grafici, schematici ed eminentemente pratici che raccolgono informazioni scientifiche e pratiche sullo Sviluppo di Software
- Novità sugli ultimi progressi nello Sviluppo di Software
- Esercizi pratici che offrono un processo di autovalutazione per migliorare l'apprendimento
- Sistema di apprendimento interattivo basato sul Metodo Casistico, e la sua applicazione alla pratica reale
- Il tutto completato da lezioni teoriche, domande all'esperto, forum di discussione su questioni controverse e compiti di riflessione individuale
- Contenuti disponibili da qualsiasi dispositivo fisso o mobile dotato di connessione a internet

Questo programma ti permetterà di conoscere la struttura di un computer e del suo software, come base per aumentare le tue competenze"

Impara tutto ciò che serve per utilizzare i linguaggi di programmazione in modo sicuro, incorporando alle tue conoscenze l'interpretazione e la progettazione di algoritmi di base per lavorare in questo campo"

Una qualifica che ti permetterà di capire il funzionamento di tutti gli elementi essenziali di un programma informatico e come intervenire su di essi.

Scopri gli ultimi sistemi di dati presenti sul mercato, impara a progettare algoritmi avanzati e tutti gli aspetti che un professionista altamente competente deve padroneggiare.

Il personale docente del programma comprende rinomati professionisti nell'ambito dello Sviluppo di Software che apportano la loro esperienza, così come specialisti appartenenti a società scientifiche di riferimento e Università prestigiose.

I contenuti multimediali, sviluppati in base alle ultime tecnologie educative, forniranno al professionista un apprendimento coinvolgente e localizzato, ovvero inserito in un contesto reale.

La progettazione di questo programma è incentrata sull'Apprendimento Basato sui Problemi, mediante il quale lo studente deve cercare di risolvere le diverse situazioni di pratica professionale che gli si presentano durante il corso. A tal fine, il professionista sarà assistito da un innovativo sistema di video interattivi creati da rinomati esperti in Sviluppo di Software che possiedono un'ampia esperienza nell'insegnamento.

02

Obiettivi

L'obiettivo di questa qualifica è quello di fornire ai professionisti che lavorano nello Sviluppo di Software le conoscenze e le competenze necessarie a svolgere la loro attività utilizzando i protocolli e le tecniche più avanzate del momento. Mediante un approccio di lavoro totalmente adattabile allo studente, questo Master Privato ti consentirà di acquisire progressivamente le competenze che ti spingeranno ad un elevato livello professionale.

Approfondirai le tue conoscenze nel campo del calcolo e della struttura dei computer, materie essenziali per qualsiasi sviluppatore di software"

Obiettivi generali

- Preparare scientificamente e tecnologicamente, nonché arricchire la pratica professionale nell'Ingegneria del Software, il tutto con una specializzazione trasversale e versatile adattata alle nuove tecnologie e alle innovazioni del settore
- Acquisirai conoscenze approfondite nel campo dell'ingegneria del software, ma anche nel campo dell'informatica e della struttura dei computer, comprese le basi matematiche, statistiche e fisiche essenziali per l'ingegneria

Raggiungi il livello di conoscenza che desideri e padroneggia la Sviluppo di Software con questa preparazione di alto livello"

Obiettivi specifici

Modulo 1. Fondamenti di programmazione

- Comprendere la struttura di base di un computer, il software e i linguaggi di programmazione di uso generale
- Imparare a progettare e interpretare gli algoritmi, che sono la base necessaria per lo sviluppo del software
- Comprendere gli elementi essenziali di un programma informatico, come i diversi tipi di dati, gli operatori, le espressioni, le dichiarazioni, le istruzioni di I/O e di controllo
- Comprendere le diverse strutture di dati disponibili nei linguaggi di programmazione generici, sia statici che dinamici, e acquisire competenze essenziali nella gestione dei file
- Comprendere le diverse tecniche di test del software e l'importanza di generare una buona documentazione insieme a un buon codice sorgente
- Imparare le basi del linguaggio di programmazione C++, uno dei più utilizzati al mondo

Modulo 2. Struttura dati

- Imparare le basi della programmazione in linguaggio C++, comprese classi, variabili, espressioni condizionali e oggetti
- Comprendere i tipi di dati astratti, i tipi di strutture dati lineari, le strutture dati gerarchiche semplici e complesse e la loro implementazione in C++
- Comprendere il funzionamento di strutture dati avanzate diverse da quelle abituali
- Comprendere la teoria e la pratica relative all'uso di heap e code prioritarie
- Imparare il funzionamento delle tabelle *Hash* come tipi di dati astratti e funzioni
- Comprendere la teoria dei grafi e dei loro concetti avanzati, nonché degli algoritmi

Modulo 3. Algoritmi e complessità

- Imparare le principali strategie per la progettazione di algoritmi, nonché i diversi metodi e misure per il loro calcolo
- Conoscere i principali algoritmi di ordinamento utilizzati nello sviluppo del software
- Capire come funzionano i diversi algoritmi su alberi, *heap* e grafi
- Comprendere il funzionamento degli algoritmi *Greedy*, la loro strategia e gli esempi del loro utilizzo nei principali problemi noti Conoscere anche l'uso degli algoritmi Greedy sui grafi
- Imparare le principali strategie di ricerca dei percorsi minimi, con l'approccio ai problemi essenziali del campo e agli algoritmi per la loro risoluzione
- Comprendere la tecnica del *Backtracking* e i suoi principali utilizzi, nonché le tecniche alternative

Modulo 4. Database

- Imparare le diverse applicazioni e finalità dei sistemi di database, nonché il loro funzionamento e la loro architettura
- Comprendere il modello relazionale, dalla sua struttura e operazioni all'algebra relazionale estesa
- Imparare in modo approfondito cosa sono i database SQL, come funzionano, come definire i dati e come creare query dalle più elementari alle più avanzate e complesse
- Imparare a progettare database utilizzando il modello entità-relazione, a creare diagrammi e a conoscere le caratteristiche del modello E-R esteso
- Approfondire la progettazione di database relazionali, analizzando le diverse forme normali e gli algoritmi di decomposizione
- Gettare le basi per comprendere il funzionamento dei database NoSQL e introdurre il database MongoDB

Modulo 5. Database avanzati

- Introdurre i diversi sistemi di database attualmente disponibili sul mercato
- Apprendere a utilizzare XML e i database per il web
- Comprendere il funzionamento di database avanzati, come i database paralleli e distribuiti
- Comprendere l'importanza dell'indicizzazione e dell'associazione nei sistemi di database
- Comprendere il funzionamento dei sistemi transazionale e di recupero
- Acquisire le conoscenze relative ai database non relazionali e al data mining

Modulo 6. Progettazione avanzata di algoritmi

- Approfondire la progettazione di algoritmi avanzati, analizzando algoritmi ricorsivi e tipo divide et impera, nonché eseguendo analisi ammortizzate
- Comprendere i concetti di programmazione dinamica e gli algoritmi per i problemi NP
- Comprendere il funzionamento dell'ottimizzazione combinatoria, i diversi algoritmi randomizzati e gli algoritmi paralleli
- Conoscere e capire come funzionano i diversi metodi di ricerca locale e con candidati
- Imparare i meccanismi della verifica formale dei programmi e di programmi iterativi, compresa la logica del primo ordine e il sistema formale di Hoare
- Imparare il funzionamento di alcuni dei principali metodi numerici come il metodo di bisezione, il metodo di Newton Raphson e il metodo della secante

Modulo 7. Interazione persona-computer

- Acquisire solide conoscenze relative all'interazione persona-computer e alla creazione di interfacce utilizzabili
- Comprendere l'importanza dell'usabilità delle applicazioni e perché è fondamentale tenerne conto nella progettazione del nostro software
- Comprendere la diversità umana, i vincoli che pone e come adattare le interfacce in base alle esigenze specifiche di ciascun tipo di diversità
- Imparare il processo di progettazione di un'interfaccia, dall'analisi dei requisiti alla valutazione, passando per le varie fasi intermedie necessarie a realizzare un'interfaccia adeguata
- Conoscere le diverse linee guida sull'accessibilità, gli standard che le stabiliscono e gli strumenti che ci permettono di valutarle
- Comprendere i diversi metodi di interazione con il computer, utilizzando periferiche e dispositivi

Modulo 8. Programmazione avanzata

- Approfondire la conoscenza della programmazione, soprattutto per quanto riguarda la programmazione orientata agli oggetti, e i diversi tipi di relazioni tra le classi esistenti
- Conoscere i diversi modelli di progettazione per i problemi orientati agli oggetti
- Imparare a conoscere la programmazione guidata dagli eventi e lo sviluppo dell'interfaccia utente con Qt
- Acquisire le conoscenze essenziali della programmazione concorrente, dei processi e dei thread
- Imparare a gestire l'uso dei thread e della sincronizzazione, nonché a risolvere i problemi più comuni della programmazione concorrente
- Comprendere l'importanza della documentazione e dei test nello sviluppo del software

Modulo 9. Sviluppo di applicazioni web

- Conoscere le caratteristiche del linguaggio di markup HTML e il suo utilizzo nella creazione di siti web insieme ai fogli di stile CSS
- Imparare a utilizzare il linguaggio di programmazione orientato al browser JavaScript e alcune delle sue caratteristiche principali
- Comprendere i concetti di programmazione orientata ai componenti e di architettura dei componenti
- Imparare a utilizzare il *Framework Frontend* Bootstrap per la progettazione di siti web
- Comprendere la struttura del controller view model nello sviluppo di siti web dinamici
- Conoscere l'architettura orientata ai servizi e le basi del protocollo HTTP

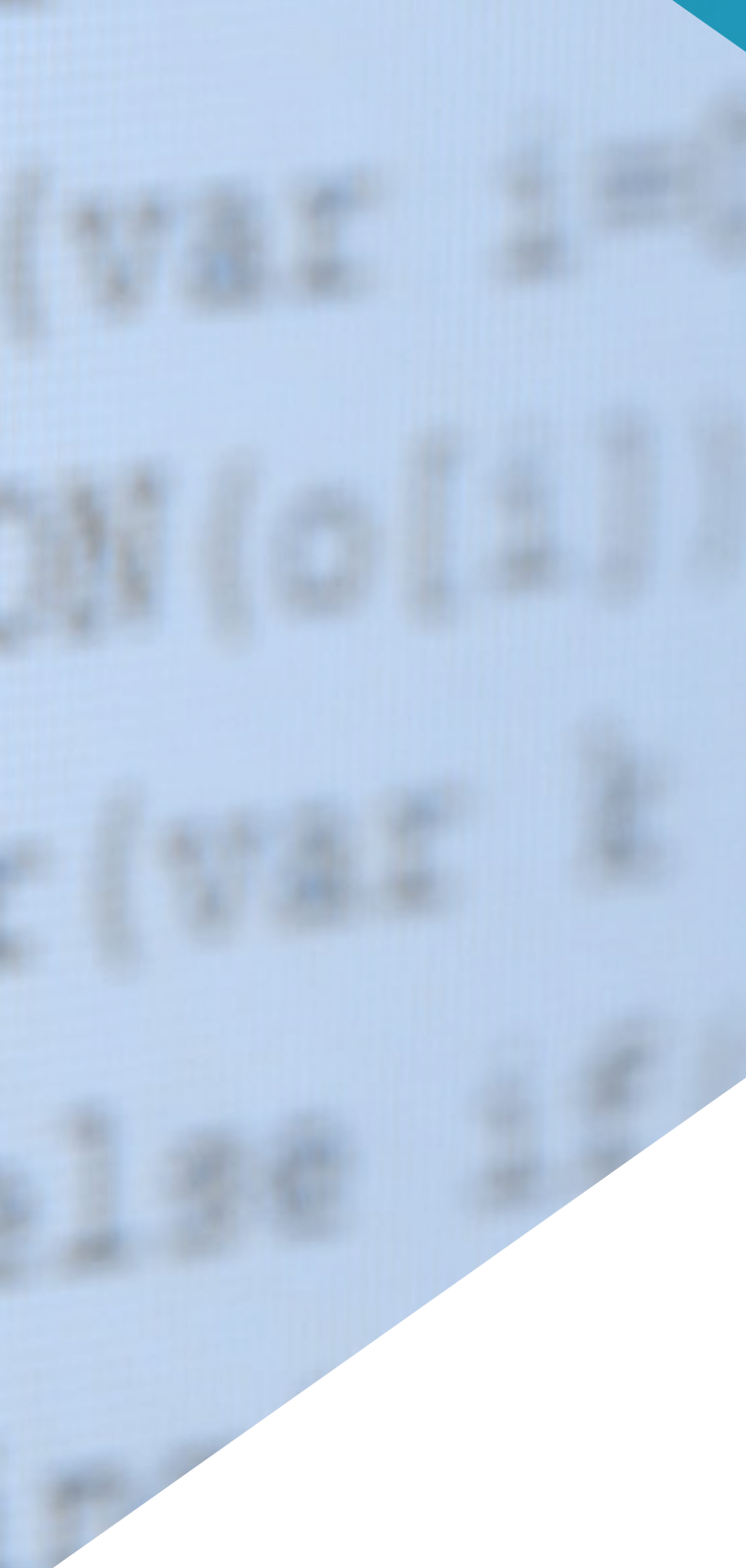
Modulo 10. Ingegneria del Software

- Gettare le basi dell'ingegneria del software e della modellazione, apprendendo i principali processi e concetti
- Comprendere il processo del software e i diversi modelli per il suo sviluppo, comprese le tecnologie agili
- Comprendere l'ingegneria dei requisiti, il loro sviluppo, elaborazione, negoziazione e validazione
- Imparare la modellazione dei requisiti e i diversi elementi come scenari, informazioni, classi di analisi, flussi, comportamenti e modelli
- Comprendere i concetti e i processi di progettazione del software, imparando anche a conoscere l'architettura di progettazione e la progettazione a livello di componenti e basata su modelli
- Conoscere i principali standard relativi alla qualità del software e alla gestione dei progetti

03

Competenze

Dopo aver superato le valutazioni del Master Privato in Sviluppo di Software, lo studente avrà acquisito le competenze necessarie per realizzare un lavoro di qualità, oltre a nuove abilità e tecniche che lo aiuteranno a completare le conoscenze di cui era già in possesso.



Aumenta le tue competenze nello Sviluppo di Software e continua a crescere come professionista in questo campo in costante evoluzione"

Competenza generale

» Rispondere alle necessità attuali dell'area dello Sviluppo di Software

Un programma eccezionale per la sua intensità, l'attualità e il modo in cui viene insegnato, che ti permetterà di fare progressi in modo rapido ed efficace"

Competenze specifiche

- Essere in grado di comprendere la struttura di base di un computer, il software e i linguaggi di programmazione di uso generale
- Saper applicare le basi della programmazione in linguaggio C++, comprese classi, variabili, espressioni condizionali e oggetti
- Conoscere in profondità le principali strategie per la progettazione di algoritmi, nonché i diversi metodi e misure per il loro calcolo
- Conoscere le diverse applicazioni e finalità dei sistemi di database, nonché il loro funzionamento e la loro architettura, e come applicarli nel lavoro quotidiano
- Essere in grado di impiegare i diversi sistemi di database attualmente disponibili sul mercato
- Saper analizzare algoritmi ricorsivi e tipo divide et impera, nonché eseguire analisi ammortizzate
- Utilizzare le conoscenze sull'interazione uomo-computer e sulla creazione di interfacce utilizzabili nella pratica quotidiana della professione
- Conoscere in maniera approfondita la programmazione
- Conoscere le caratteristiche del linguaggio di markup HTML e il suo utilizzo nella creazione di siti web insieme ai fogli di stile CSS
- Essere in grado di applicare i principali processi e concetti delle basi dell'ingegneria del software e della modellazione

04

Struttura e contenuti

La struttura dei contenuti è stata progettata da una squadra di professionisti dell'Ingegneria Informatica con l'obiettivo di garantire agli studenti del Master Privato un apprendimento efficiente e rapido. A tal fine, i contenuti sono stati organizzati in modo che l'apprendimento sia intenso e costante, cercando di mantenere la motivazione in base ai progressi costanti dello studente.

Un programma educativo finalizzato al raggiungimento di competenze complete nello sviluppo del software, che ti proietterà a un livello professionale superiore"

Modulo 1. Fondamenti di programmazione

1.1. Introduzione alla programmazione
- 1.1.1. Struttura base di un computer
- 1.1.2. Software
- 1.1.3. Linguaggi di programmazione
- 1.1.4. Ciclo di vita di un'applicazione informatica

1.2. Progettazione di algoritmi
- 1.2.1. Risoluzione dei problemi
- 1.2.2. Tecniche descrittive
- 1.2.3. Elementi e struttura di un algoritmo

1.3. Elementi di un programma
- 1.3.1. Origine e caratteristiche del linguaggio C++
- 1.3.2. Ambienti di sviluppo
- 1.3.3. Concetto di programma
- 1.3.4. Tipi di dati fondamentali
- 1.3.5. Operatori
- 1.3.6. Espressioni
- 1.3.7. Strutture
- 1.3.8. Ingresso e uscita di dati

1.4. Strutture di controllo
- 1.4.1. Strutture
- 1.4.2. Biforcazioni
- 1.4.3. Loop

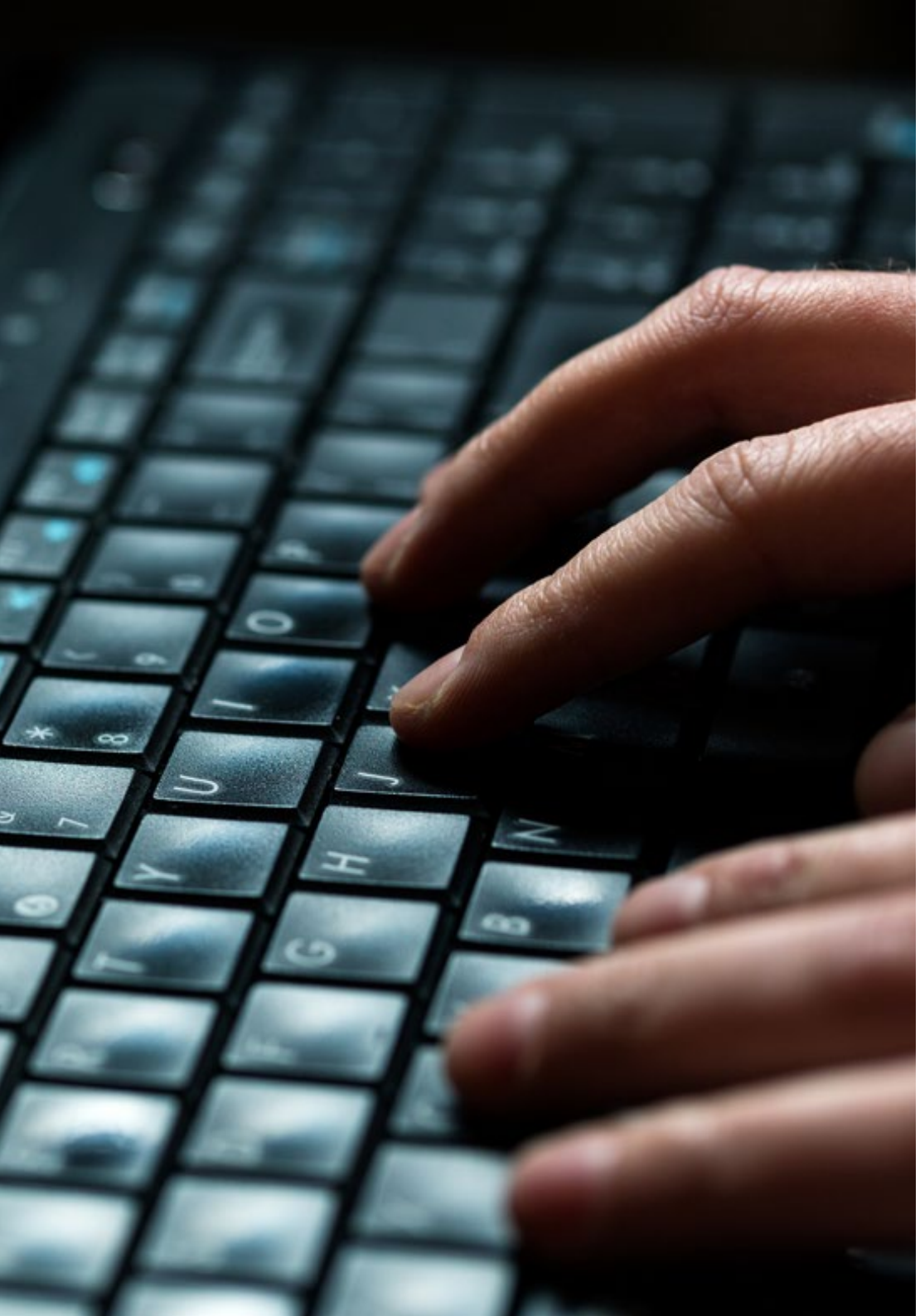
1.5. Astrazione e modularità: le funzioni
- 1.5.1. Progettazione modulare
- 1.5.2. Concetto di funzione e utilità
- 1.5.3. Definizione di una funzione
- 1.5.4. Flusso di esecuzione in una chiamata di funzione
- 1.5.5. Prototipo di una funzione
- 1.5.6. Restituzione dei risultati
- 1.5.7. Chiamata di una funzione: parametri
- 1.5.8. Passaggio di parametri per riferimento e per valore
- 1.5.9. Ambito di identificazione

1.6. Strutture dati statiche
- 1.6.1. *Array*
- 1.6.2. Matrici. Poliedri
- 1.6.3. Ricerca e ordinamento
- 1.6.4. Stringhe. Funzioni di I/O per le catene
- 1.6.5. Strutture. Unioni
- 1.6.6. Nuovi tipi di dati

1.7. Strutture dati dinamiche: puntatori
- 1.7.1. Concetto. Definizione di puntatore
- 1.7.2. Operatori e operazioni con i puntatori
- 1.7.3. *Array* di puntatori
- 1.7.4. Puntatori e *array*
- 1.7.5. Puntatori a stringhe
- 1.7.6. Puntatori a strutture
- 1.7.7. Indirezione multipla
- 1.7.8. Puntatori a funzioni
- 1.7.9. Passaggio di funzioni, strutture e *array* come parametri di funzione

1.8. Cartelle
- 1.8.1. Concetti di base
- 1.8.2. Operazioni sui file
- 1.8.3. Tipi di cartelle
- 1.8.4. Organizzazione dei file
- 1.8.5. Introduzione ai file C++
- 1.8.6. Gestione dei file

1.9. Ricorsione
- 1.9.1. Definizione di ricorsione
- 1.9.2. Tipi di ricorsione
- 1.9.3. Vantaggi e svantaggi
- 1.9.4. Considerazioni
- 1.9.5. Conversione ricorsiva iterativa
- 1.9.6. Lo stack di ricorsione



1.10. Prove e documentazione
- 1.10.1. Test del programma
- 1.10.2. Test della scatola bianca
- 1.10.3. Test della scatola nera
- 1.10.4. Strumenti per i test
- 1.10.5. Documentazione del programma

Modulo 2. Struttura dati

2..1. Introduzione alla programmazione C++
- 2.1.1. Classi, costruttori, metodi e attributi
- 2.1.2. Variabili
- 2.1.3. Espressioni condizionali e loop
- 2.1.4. Obiettivi

2.2. Tipi di dati astratti (ADT)
- 2.2.1. Tipi di dati
- 2.2.2. Strutture di base e ADT
- 2.2.3. Vettori e *array*

2.3. Strutture dati lineari
- 2.3.1. Lista ADT, definizione
- 2.3.2. Liste collegate e doppiamente collegate
- 2.3.3. Liste ordinate
- 2.3.4. Liste in C++
- 2.3.5. ADT pila
- 2.3.6. ADT coda
- 2.3.7. Pila e coda in C++

2.4. Strutture dati gerarchiche
- 2.4.1. ADT albero
- 2.4.2. Percorsi
- 2.4.3. Alberi n-ari
- 2.4.4. Alberi binari
- 2.4.5. Alberi binari di ricerca

2.5. Strutture dati gerarchiche: alberi complessi
  2.5.1. Alberi perfettamente bilanciati o di altezza minima
  2.5.2. Alberi multipercorso
  2.5.3. Riferimenti bibliografici
2.6. Heap e code di priorità
  2.6.1. ADT heap
  2.6.2. ADT coda di priorità
2.7. Tabelle *Hash*
  2.7.1. TAD Tabella *Hash*
  2.7.2. Funzioni *Hash*
  2.7.3. Funzioni *Hash* nelle tabelle *Hash*
  2.7.4. Ridispersione
  2.7.5. Tabelle *Hash* aperte
2.8. Grafi
  2.8.1. ADT grafo
  2.8.2. Tipi di grafo
  2.8.3. Rappresentazione grafica e operazioni di base
  2.8.4. Progettazione di grafi
2.9. Algoritmi e concetti avanzati sui grafi
  2.9.1. Problemi sui grafi
  2.9.2. Algoritmi sui percorsi
  2.9.3. Algoritmi di ricerca o percorsi
  2.9.4. Altri algoritmi
2.10. Altre strutture di dati
  2.10.1. Insiemi
  2.10.2. *Array* paralleli
  2.10.3. Tabelle di simboli
  2.10.4. *Trie*

Modulo 3. Algoritmi e complessità

3.1. Introduzione alle strategie di progettazione degli algoritmi
  3.1.1. Ricorsione
  3.1.2. Divide et impera
  3.1.3. Altre strategie
3.2. Efficienza e analisi degli algoritmi
  3.2.1. Misure di efficienza
  3.2.2. Misurare le dimensioni dell'ingresso
  3.2.3. Misurare il tempo di esecuzione
  3.2.4. Caso peggiore, caso migliore e caso medio
  3.2.5. Notazione asintotica
  3.2.6. Criteri di analisi matematica per algoritmi non ricorsivi
  3.2.7. Analisi matematica di algoritmi ricorsivi
  3.2.8. Analisi empirica degli algoritmi
3.3. Algoritmi di ordinamento
  3.3.1. Concetto di gestione
  3.3.2. Ordinamento per bolla
  3.3.3. Ordinamento per selezione
  3.3.4. Ordinamento per inserimento
  3.3.5. Merge Sort
  3.3.6. QuickSort
3.4. Algoritmi con alberi
  3.4.1. Concetto di albero
  3.4.2. Alberi binari
  3.4.3. Percorsi di alberi
  3.4.4. Rappresentare espressioni
  3.4.5. Alberi binari ordinati
  3.4.6. Alberi binari bilanciati

3.5. Algoritmi con *Heap*
  3.5.1. Gli *Heap*
  3.5.2. L'algoritmo HeapSort
  3.5.3. Code di priorità
3.6. Algoritmi con grafi
  3.6.1. Rappresentazione
  3.6.2. Percorso in larghezza
  3.6.3. Percorso in profondità
  3.6.4. Ordinamento topologico
3.7. Algoritmi *Greedy*
  3.7.1. La strategia *Greedy*
  3.7.2. Elementi della strategia *Greedy*
  3.7.3. Scambio di monete
  3.7.4. Problema del viaggiatore
  3.7.5. Problema dello zaino
3.8. Ricerca di percorsi minimi
  3.8.1. Il problema del percorso minimo
  3.8.2. Archi negativi e cicli
  3.8.3. Algoritmo di Dijkstra
3.9. Algoritmi *greedy* sui grafi
  3.9.1. L'albero di copertura minimo
  3.9.2. Algoritmo di Prim
  3.9.3. Algoritmo di Kruskal
  3.9.4. Analisi della complessità
3.10. *Backtracking*
  3.10.1. Il *Backtracking*
  3.10.2. Tecniche alternative

Modulo 4. Database

4.1. Applicazioni e finalità dei sistemi di database
  4.1.1. Applicazioni dei diversi sistemi di database
  4.1.2. Protezione dei diversi sistemi di database
  4.1.3. Il punto di vista dei dati
4.2. Database e architettura
  4.2.1. Database relazionali
  4.2.2. Progettazione di database
  4.2.3. Database a oggetti e semi-strutturati
  4.2.4. Memorizzazione di dati e consultazioni
  4.2.5. Gestione delle transazioni
  4.2.6. Mining e analisi dati
  4.2.7. Architettura sui database
4.3. Il modello relazionale: struttura, operazioni e algebra relazionale estesa
  4.3.1. La struttura dei database relazionali
  4.3.2. Operazioni fondamentali dell'algebra relazionale
  4.3.3. Altre operazioni di algebra relazionale
  4.3.4. Operazioni dell'algebra relazionale estesa
  4.3.5. Valori nulli
  4.3.6. Modifica di database
4.4. SQL (I)
  4.4.1. Cos'è il SQL?
  4.4.2. Definizione dei dati
  4.4.3. Struttura di base delle query SQL
  4.4.4. Operazioni sugli insiemi
  4.4.5. Funzioni di aggregazione
  4.4.6. Valori nulli

4.5. SQL (II)
- 4.5.1. Sub-query annidate
- 4.5.2. Query complesse
- 4.5.3. Viste
- 4.5.4. Cursori
- 4.5.5. Query complesse
- 4.5.6. Scatti

4.6. Progettazione di database e modello E-R
- 4.6.1. Panoramica del processo di progettazione
- 4.6.2. Modello entità relazione
- 4.6.3. Restrizioni

4.7. Diagramma entità relazione
- 4.7.1. Diagramma entità relazione
- 4.7.2. Aspetti della progettazione entità relazione
- 4.7.3. Insiemi di entità deboli

4.8. Mod.ello entità relazione esteso
- 4.8.1. Caratteristiche del modello E-R esteso
- 4.8.2. Progettazione di database
- 4.8.3. Riduzione a schemi relazionali

4.9. Progettazione di database relazionali
- 4.9.1. Caratteristiche di un buon progetto relazionale
- 4.9.2. Domini atomici e prima forma normale (1FN)
- 4.9.3. Decomposizione attraverso le dipendenze funzionali
- 4.9.4. Teoria della dipendenza funzionale
- 4.9.5. Algoritmi di decomposizione
- 4.9.6. Decomposizione tramite le dipendenze funzionali
- 4.9.7. Altre forme normali
- 4.9.8. Processo di progettazione di database

4.10. Database NoSQL
- 4.10.1. Cosa sono i database NoSQL?
- 4.10.2. Analisi delle diverse opzioni NoSQL e delle loro caratteristiche
- 4.10.3. MongoDB

Modulo 5. Database avanzati

5.1. Introduzione ai diversi sistemi di database
- 5.1.1. Rassegna storica
- 5.1.2. Database gerarchici
- 5.1.3. Database web
- 5.1.4. Database relazionale
- 5.1.5. Database non relazionale

5.2. XML e database per il web
- 5.2.1. Convalida dei documenti XML
- 5.2.2. Trasformazioni dei documenti XML
- 5.2.3. Memorizzazione di dati XML
- 5.2.4. Database relazionale XML
- 5.2.5. SQL/XML
- 5.2.6. Database native XML

5.3. Database paralleli
- 5.3.1. Sistemi paralleli
- 5.3.2. Architetture parallele di database
- 5.3.4. Parallelismo nelle consultazioni
- 5.3.5. Parallelismo tra consultazioni
- 5.3.6. Progettazione di sistemi paralleli
- 5.3.7. Elaborazione parallela in SQL

5.4. Database distribuiti
- 5.4.1. Sistemi distribuiti
- 5.4.2. Memorizzazione distribuita
- 5.4.3. Disponibilità
- 5.4.4. Elaborazione distribuita delle query
- 5.4.5. Fornitori di database distribuiti

5.5. Indicizzazione e associazione
- 5.5.1. Indici ordinati
- 5.5.2. Indici densi e sparsi
- 5.5.3. Indici multilivello
- 5.5.4. Aggiornamento dell'indice
- 5.5.5. Associazione statica
- 5.5.6. Come utilizzare gli indici nei database

5.6. Introduzione all'elaborazione transazionale
- 5.6.1. Stati di una transazione
- 5.6.2. Implementazione dell'atomicità e della durata
- 5.6.3. Sequenzialità
- 5.6.4. Recuperabilità
- 5.6.5. Implementazione dell'isolamento

5.7. Sistemi di recupero
- 5.7.1. Classificazione dei guasti
- 5.7.2. Strutture di stoccaggio
- 5.7.3. Recupero e atomicità
- 5.7.4. Recupero basato sui dati storici
- 5.7.5. Transazioni concomitanti e recupero
- 5.7.6. Alta disponibilità nei database

5.8. Esecuzione ed elaborazione di query
- 5.8.1. Costi di un query
- 5.8.2. Operazione di selezione
- 5.8.3. Ordinamento
- 5.8.4. Introduzione all'ottimizzazione delle query
- 5.8.5. Monitoraggio delle prestazioni

5.9. Database non relazionale
- 5.9.1. Database orientati ai documenti
- 5.9.2. Database orientati ai grafi
- 5.9.3. Database a valore-chiave

5.10. Data *Warehouse*, OLAP e data mining
- 5.10.1. Componenti dei data warehouse
- 5.10.2. Architettura dei data *Warehouse*
- 5.10.3. OLAP
- 5.10.4. Funzionalità del data mining
- 5.10.5. Altri tipi di mining

Modulo 6. Progettazione avanzata di algoritmi

6.1. Analisi di algoritmi ricorsivi e divide et impera
- 6.1.1. Porre e risolvere equazioni di ricorrenza omogenee e non omogenee
- 6.1.2. Panoramica della strategia divide et impera

6.2. Analisi ammortizzata
- 6.2.1. Analisi aggregata
- 6.2.2. Il metodo di contabilità
- 6.2.3. Il metodo del potenziale

6.3. Programmazione dinamica e algoritmi per problemi NP
- 6.3.1. Caratteristiche della programmazione dinamica
- 6.3.2. Indietro nel tempo: *Backtracking*
- 6.3.3. Ramificazione e potatura

6.4. Ottimizzazione combinatoria
- 6.4.1. Rappresentazione dei problemi
- 6.4.2. Ottimizzazione 1D

6.5. Algoritmi randomizzati
- 6.5.1. Esempi di algoritmi randomizzati
- 6.5.2. Il teorema Buffon
- 6.5.3. Algoritmo Monte Carlo
- 6.5.4. Algoritmo Las Vegas

6.6. Ricerca locale e con candidati
- 6.6.1. *Gradient Ascent*
- 6.6.2. *Hill Climbing*
- 6.6.3. *Simulated Annealing*
- 6.6.4. *Tabu Search*
- 6.6.5. Ricerca con candidati

6.7. Verifica formale dei programmi
- 6.7.1. Specifica delle astrazioni funzionali
- 6.7.2. Il linguaggio della logica di prim'ordine
- 6.7.3. Sistema formale di Hoare

6.8. Verifica dei programmi iterativi
- 6.8.1. Regole del sistema formale di Hoare
- 6.8.2. Concetto invariante di iterazioni

6.9. Metodi numerici
- 6.9.1. Il metodo della bisezione
- 6.9.2. Metodo di Newton Raphson
- 6.9.3. Il metodo delle secanti

6.10. Algoritmi paralleli
- 6.10.1. Operazioni binarie parallele
- 6.10.2. Operazioni parallele con grafi
- 6.10.3. Parallelismo nel divide et impera
- 6.10.4. Parallelismo nella programmazione dinamica

Modulo 7. Interazione persona-computer

7.1. Introduzione all'interazione persona-computer
- 7.1.1. Cos'è l'interazione persona-computer
- 7.1.2. Relazione dell'interazione persona-computer con altre discipline
- 7.1.3. Interfaccia utente
- 7.1.4. Usabilità e accessibilità
- 7.1.5. Esperienza utente e design incentrato sull'utente

7.2. Il computer e l'interazione: interfaccia utente e paradigmi di interazione
- 7.2.1. Interazione
- 7.2.2. Paradigmi e stili di interazione
- 7.2.3. Evoluzione dell'interfaccia utente
- 7.2.4. Interfacce utente classiche: WIMP/GUI, comandi, voce, realtà virtuale
- 7.2.5. Interfacce utente innovative: mobile, portatile, collaborative, BCI

7.3. Il fattore umano: aspetti psicologici e cognitivi
- 7.3.1. L'importanza del fattore umano nell'interazione
- 7.3.2. Il processo umano di informazione
- 7.3.3. L'ingresso e l'uscita delle informazioni: visive, uditive e tattili
- 7.3.4. Percezione e attenzione
- 7.3.5. Conoscenza e modelli mentali: rappresentazione, organizzazione e acquisizione

7.4. Il fattore umano: limitazioni sensoriali e fisiche
- 7.4.1. Diversità funzionale, disabilità e menomazione
- 7.4.2. Diversità visiva
- 7.4.3. Diversità auditiva
- 7.4.4. Diversità cognitiva
- 7.4.5. Diversità motoria
- 7.4.6. Il caso dei migranti digitali

7.5. Il processo di progettazione (I): analisi dei requisiti per la progettazione dell'interfaccia utente
- 7.5.1. Progettazione incentrata sull'utente
- 7.5.2. Difficoltà nell'analisi dei requisiti
- 7.5.3. Raccolta di informazioni
- 7.5.4. Analisi e interpretazione dell'informazione
- 7.5.5. Analisi di usabilità e accessibilità

7.6. Il processo di progettazione (II): prototipazione e analisi dei compiti
- 7.6.1. Progettazione concettuale
- 7.6.2. Prototipazione
- 7.6.3. Analisi gerarchica dei compiti

7.7. Il processo di progettazione (III): la valutazione
- 7.7.1. La valutazione nel processo di progettazione: obiettivi e metodi
- 7.7.2. Metodi di valutazione senza utenti
- 7.7.3. Metodi di valutazione con gli utenti
- 7.7.4. Standard e norme di valutazione

7.8. Accessibilità: definizione e linee guida
- 7.8.1. Accessibilità e progettazione universale
- 7.8.2. L'iniziativa WAI e le linee guida WCAG
- 7.8.3. Linee guida WCAG 2.0. e 2.1

7.9. Accessibilità: valutazione e diversità funzionale
- 7.9.1. Strumenti per la valutazione dell'accessibilità dei siti web
- 7.9.2. Accessibilità e diversità funzionale

7.10. Il computer e l'interazione: periferiche e dispositivi
- 7.10.1. Dispositivi e periferiche tradizionali
- 7.10.2. Dispositivi e periferiche alternative
- 7.10.3. Telefoni cellulari e tablet
- 7.10.4. Diversità funzionale, interazione e periferiche

Modulo 8. Programmazione avanzata

8.1. Introduzione alla programmazione orientata agli oggetti
- 8.1.1. Introduzione alla programmazione orientata agli oggetti
- 8.1.2. Progettazione di classi
- 8.1.3. Introduzione a UML per la modellazione dei problemi

8.2. Relazioni tra classi
- 8.2.1. Astrazione ed ereditarietà
- 8.2.2. Concetti avanzati di ereditarietà
- 8.2.3. Polimorfismi
- 8.2.4. Composizione e aggregazione

8.3. Introduzione ai modelli di progettazione per i problemi orientati agli oggetti
- 8.3.1. Cosa sono i modelli di progettazione
- 8.3.2. Modello *Factory*
- 8.3.4. Modello *Singleton*
- 8.3.5. Modello *Observer*
- 8.3.6. Modello *Composite*

8.4. Eccezioni
- 8.4.1. Cosa sono le eccezioni?
- 8.4.2. Cattura e gestione delle eccezioni
- 8.4.3. Avvio delle eccezioni
- 8.4.4. Creazione di eccezioni

8.5. Interfacce utente
- 8.5.1. Introduzione a Qt
- 8.5.2. Posizionamento
- 8.5.3. Cosa sono gli eventi?
- 8.5.4. Eventi: definizione e cattura
- 8.5.5. Sviluppo dell'interfaccia utente

8.6. Introduzione alla programmazione concorrente
- 8.6.1. Introduzione alla programmazione concorrente
- 8.6.2. Il concetto di processo e di thread
- 8.6.3. Interazione tra processi o thread
- 8.6.4. Thread in C++
- 8.6.6. Vantaggi e svantaggi della programmazione concorrente

8.7. Gestione e sincronizzazione dei thread
- 8.7.1. Ciclo di vita dei thread
- 8.7.2. La classe *Thread*
- 8.7.3. Pianificazione dei thread
- 8.7.4. Gruppi di thread
- 8.7.5. Thread daemon
- 8.7.6. Sincronizzazione
- 8.7.7. Meccanismi di bloccaggio
- 8.7.8. Meccanismi di comunicazione
- 8.7.9. Monitor

8.8. Problemi comuni della programmazione concorrente
- 8.8.1. Il problema dei produttori-consumatori
- 8.8.2. Il problema dei lettori e degli scrittori
- 8.8.3. Il problema della cena dei filosofi

8.9. Documentazione e test del software
- 8.9.1. Perché è importante documentare il software?
- 8.9.2. Documento di progettazione
- 8.9.3. Utilizzo di strumenti per la documentazione

8.10. Test del software
- 8.10.1. Introduzione ai test del software
- 8.10.2. Tipi di test
- 8.10.3. Test unitario
- 8.10.4. Test di integrazione
- 8.10.5. Test di validazione
- 8.10.6. Test del sistema

Modulo 9. Sviluppo di applicazioni web

9.1. Linguaggi di markup HTML5
- 9.1.1. Concetti base di HTML
- 9.1.2. Nuovi elementi HTML 5
- 9.1.3. Formulari: nuovi controlli

9.2. Introduzione ai fogli di stile CSS
- 9.2.1. Primi passi con i CSS
- 9.2.2. Introduzione ai CSS3

9.3. Linguaggio di scripting del browser: JavaScript
- 9.3.1. Concetti base di JavaScript
- 9.3.2. DOM
- 9.3.3. Eventi
- 9.3.4. JQuery
- 9.3.5. Ajax

9.4. Concetto di programmazione orientata ai componenti
- 9.4.1. Contesto
- 9.4.2. Componenti e interfacce
- 9.4.3. Stati di un componente

9.5. Architettura dei componenti
- 9.5.1. Architetture attuali
- 9.5.2. Integrazione e distribuzione dei componenti

9.6. *Framework Frontend*: Bootstrap
- 9.6.1. Progettazione con griglia
- 9.6.2. Moduli
- 9.6.3. Componenti

9.7. Model view controller
- 9.7.1. Metodi di sviluppo Web
- 9.7.2. Modelli di disegno: MVC

9.8. Tecnologie informatiche grid
- 9.8.1. Aumento delle risorse informatiche
- 9.8.2. Concetto di tecnologia grid

9.9. Architettura orientata ai servizi
- 9.9.1. SOA e servizi web
- 9.9.2. Topologia di un servizio web
- 9.9.3. Piattaforme per servizi web

9.10. Protocollo HTTP
- 9.10.1. Messaggi
- 9.10.2. Sessioni persistenti
- 9.10.3. Sistemi crittografici
- 9.10.4. Funzionamento del protocollo HTTPS

Modulo 10. Ingegneria del Software

10.1. Introduzione all'ingegneria del software e alla modellazione
- 10.1.1. La natura del software
- 10.1.2. La natura unica delle WebApp
- 10.1.3. Ingegneria del software
- 10.1.4. Il processo del software
- 10.1.5. La pratica dell'ingegneria del software
- 10.1.6. Miti del software
- 10.1.7. Come tutto ebbe inizio
- 10.1.8. Concetti orientati agli oggetti
- 10.1.9. Introduzione a UML

10.2. Il processo del software
- 10.2.1. Un modello generale di processo
- 10.2.2. Modelli di processo prescrittivi
- 10.2.3. Modelli di processo specializzato
- 10.2.4. Il processo unificato
- 10.2.5. Modelli di processo personali e di gruppo
- 10.2.6. Che cos'è l'agilità?
- 10.2.7. Che cos'è un processo agile?
- 10.2.8. Scrum
- 10.2.9. Kit di strumenti per i processi agili

10.3. Principi che guidano la pratica dell'ingegneria del software
- 10.3.1. Principi che guidano il processo
- 10.3.2. Principi che guidano la pratica
- 10.3.3. Principi di comunicazione
- 10.3.4. Principi di pianificazione
- 10.3.5. Principi di modellazione
- 10.3.6. Principi di costruzione
- 10.3.7. Principi di implementazione

10.4. Comprendere i requisiti
- 10.4.1. Ingegneria dei requisiti
- 10.4.2. Gettare le basi
- 10.4.3. Indagare i requisiti
- 10.4.4. Sviluppo di casi d'uso
- 10.4.5. Elaborazione del modello dei requisiti
- 10.4.6. Negoziazione dei requisiti
- 10.4.7. Convalida dei requisiti

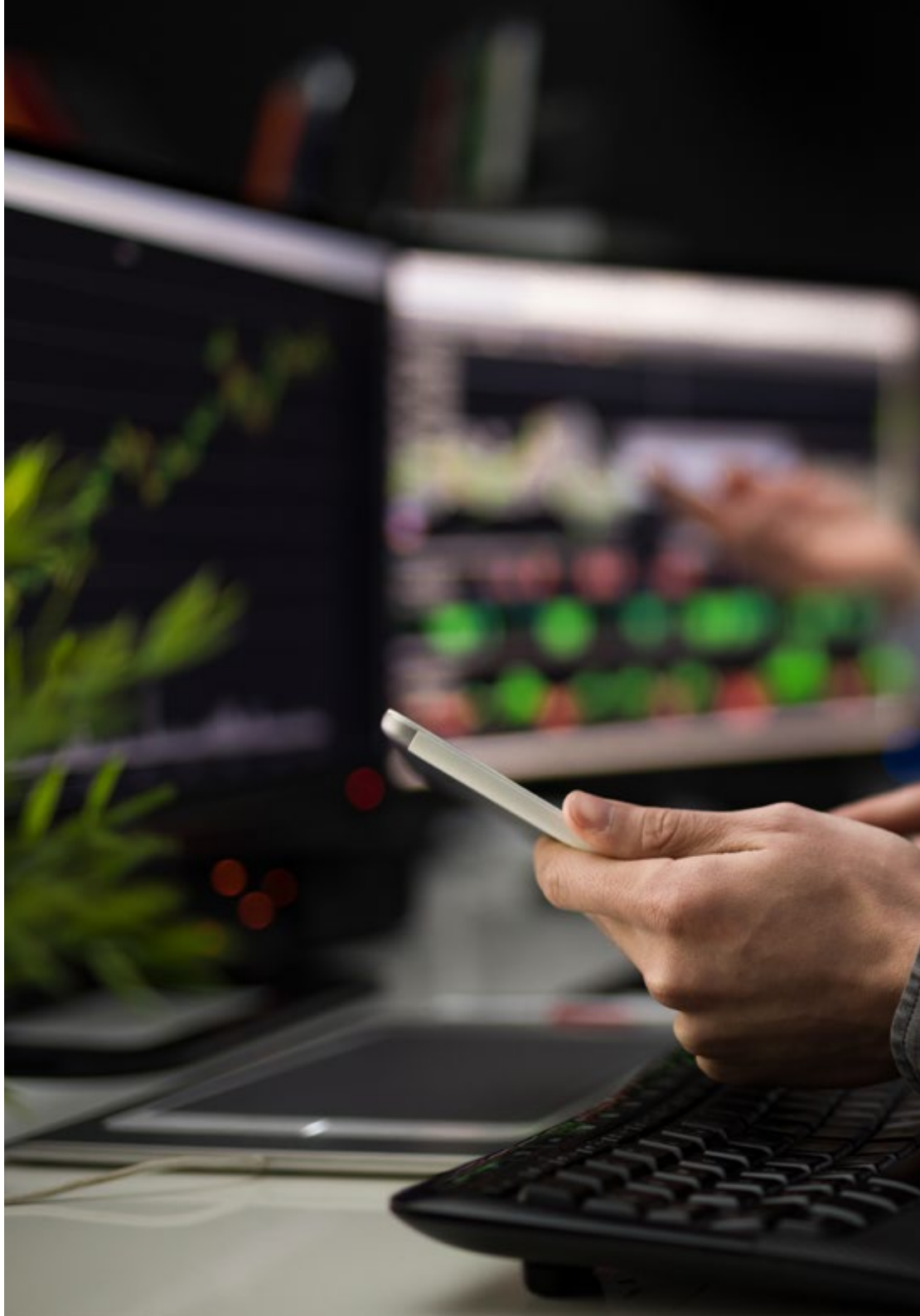
10.5. Modellazione dei requisiti: scenari, informazioni e classi di analisi
- 10.5.1. Analisi dei requisiti
- 10.5.2. Modellazione basata su scenari
- 10.5.3. Modelli UML che forniscono il caso d'uso
- 10.5.4. Concetti di modellazione dei dati
- 10.5.5. Modellazione basata su classi
- 10.5.6. Diagrammi di classe

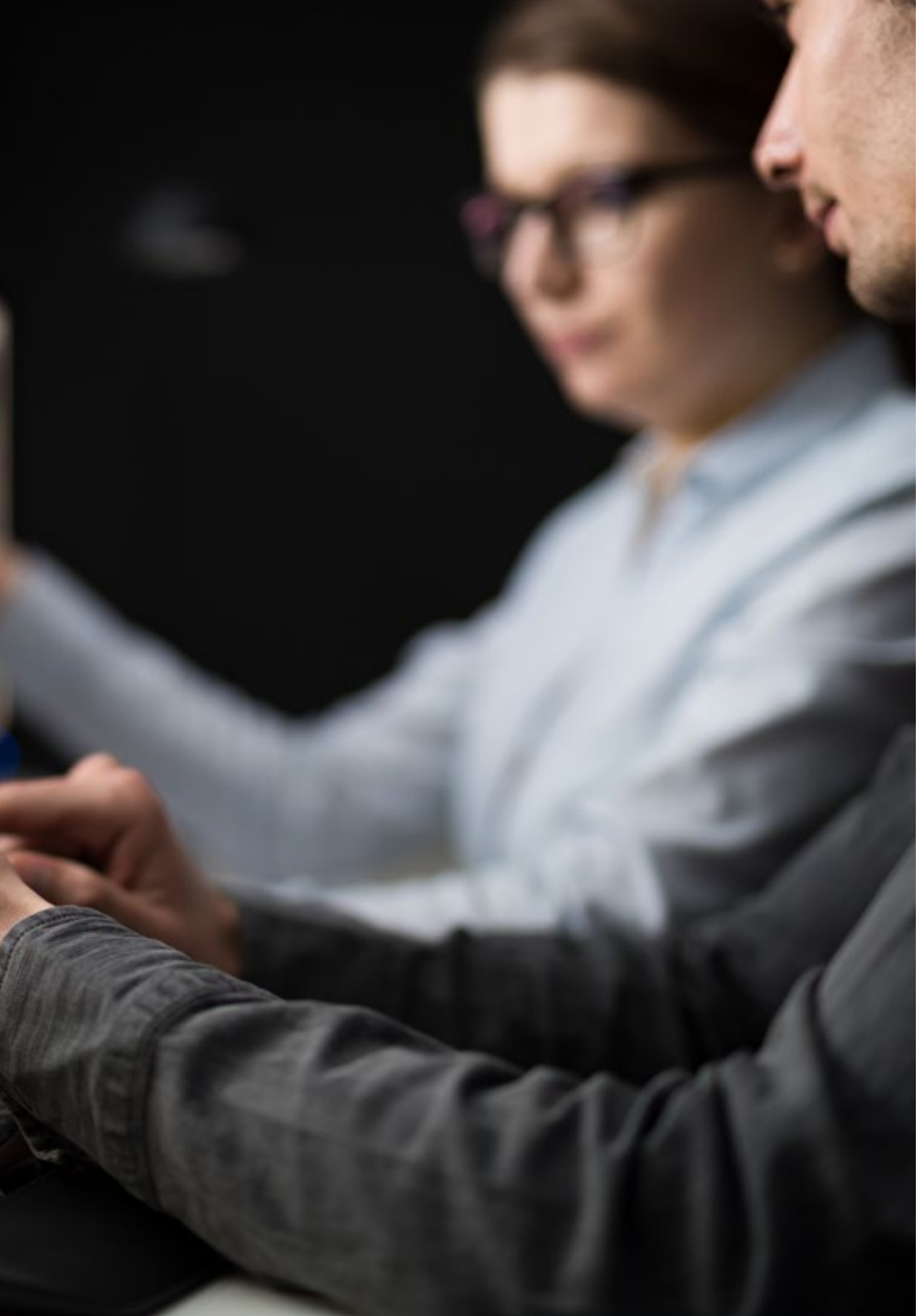
10.6. Modellazione dei requisiti: flusso, comportamento e modelli
- 10.6.1. Strategie di definizione dei requisiti
- 10.6.2. Modellazione orientata al flusso
- 10.6.3. Diagrammi di stato
- 10.6.4. Creazione di una modellazione del comportamento
- 10.6.5. Diagrammi di sequenza
- 10.6.6. Diagrammi di comunicazione
- 10.6.7. Linee guida per la modellazione dei requisiti

10.7. Concetti di progettazione
- 10.7.1. La progettazione nel contesto dell'ingegneria del software
- 10.7.2. Il processo di progettazione
- 10.7.3. Concetti di progettazione
- 10.7.4. Concetti di progettazione orientati agli oggetti
- 10.7.5. Il modello di progettazione

10.8. Progettazione dell'architettura
- 10.8.1. Architettura del software
- 10.8.2. Generi architettonici
- 10.8.3. Stili architettonici
- 10.8.4. Progettazione architettonica
- 10.8.5. Evoluzione dei progetti alternativi per l'architettura
- 10.8.6. Mappatura dell'architettura con l'uso di flussi di dati

10.9. Progettazione a livello di componente e basata su modelli
- 10.9.1. Che cos'è un componente?
- 10.9.2. Progettazione di componenti basati su classi
- 10.9.3. Realizzazione del progetto a livello di componenti
- 10.9.4. Progettazione di componenti tradizionali
- 10.9.5. Sviluppo basato su componenti
- 10.9.6. Modelli di disegno
- 10.9.7. Progettazione di software basato su modelli
- 10.9.8. Modelli architettonici
- 10.9.9. Modelli di progettazione a livello di componenti
- 10.9.10. Modelli di progettazione dell'interfaccia utente





10.10. Qualità del software e gestione dei progetti
- 10.10.1. Qualità
- 10.10.1. Qualità del software
- 10.10.2. Il dilemma della qualità del software
- 10.10.3. Raggiungere la qualità del software
- 10.10.4. Garanzie della qualità del software
- 10.10.5. Lo spettro amministrativo
- 10.10.6. Il personale
- 10.10.7. Il prodotto
- 10.10.8. Il processo
- 10.10.9. Il progetto
- 10.10.10. Principi e pratiche

Un'esperienza formativa unica, chiave e decisiva per potenziare il tuo sviluppo professionale"

05

Metodologia

Questo programma ti offre un modo differente di imparare. La nostra metodologia si sviluppa in una modalità di apprendimento ciclico: ***il Relearning***.

Questo sistema di insegnamento viene applicato nelle più prestigiose facoltà di medicina del mondo ed è considerato uno dei più efficaci da importanti pubblicazioni come il ***New England Journal of Medicine.***

Scopri il Relearning, un sistema che abbandona l'apprendimento lineare convenzionale, per guidarti attraverso dei sistemi di insegnamento ciclici: una modalità di apprendimento che ha dimostrato la sua enorme efficacia, soprattutto nelle materie che richiedono la memorizzazione"

Caso di Studio per contestualizzare tutti i contenuti

Il nostro programma offre un metodo rivoluzionario per sviluppare le abilità e le conoscenze. Il nostro obiettivo è quello di rafforzare le competenze in un contesto mutevole, competitivo e altamente esigente.

Con TECH potrai sperimentare un modo di imparare che sta scuotendo le fondamenta delle università tradizionali in tutto il mondo"

Avrai accesso a un sistema di apprendimento basato sulla ripetizione, con un insegnamento naturale e progressivo durante tutto il programma.

Imparerai, attraverso attività collaborative e casi reali, la risoluzione di situazioni complesse in ambienti aziendali reali.

Un metodo di apprendimento innovativo e differente

Questo programma di TECH consiste in un insegnamento intensivo, creato ex novo, che propone le sfide e le decisioni più impegnative in questo campo, sia a livello nazionale che internazionale. Grazie a questa metodologia, la crescita personale e professionale viene potenziata, effettuando un passo decisivo verso il successo. Il metodo casistico, la tecnica che sta alla base di questi contenuti, garantisce il rispetto della realtà economica, sociale e professionale più attuali.

> *Il nostro programma ti prepara ad affrontare nuove sfide in ambienti incerti e a raggiungere il successo nella tua carriera"*

Il Metodo Casistico è stato il sistema di apprendimento più usato nelle migliori Scuole di Informatica del mondo da quando esistono. Sviluppato nel 1912 affinché gli studenti di Diritto non imparassero la legge solo sulla base del contenuto teorico, il metodo casistico consisteva nel presentare loro situazioni reali e complesse per prendere decisioni informate e giudizi di valore su come risolverle. Nel 1924 fu stabilito come metodo di insegnamento standard ad Harvard.

Cosa dovrebbe fare un professionista per affrontare una determinata situazione? Questa è la domanda con cui ti confrontiamo nel metodo dei casi, un metodo di apprendimento orientato all'azione. Durante il corso, gli studenti si confronteranno con diversi casi di vita reale. Dovranno integrare tutte le loro conoscenze, effettuare ricerche, argomentare e difendere le proprie idee e decisioni.

Metodologia Relearning

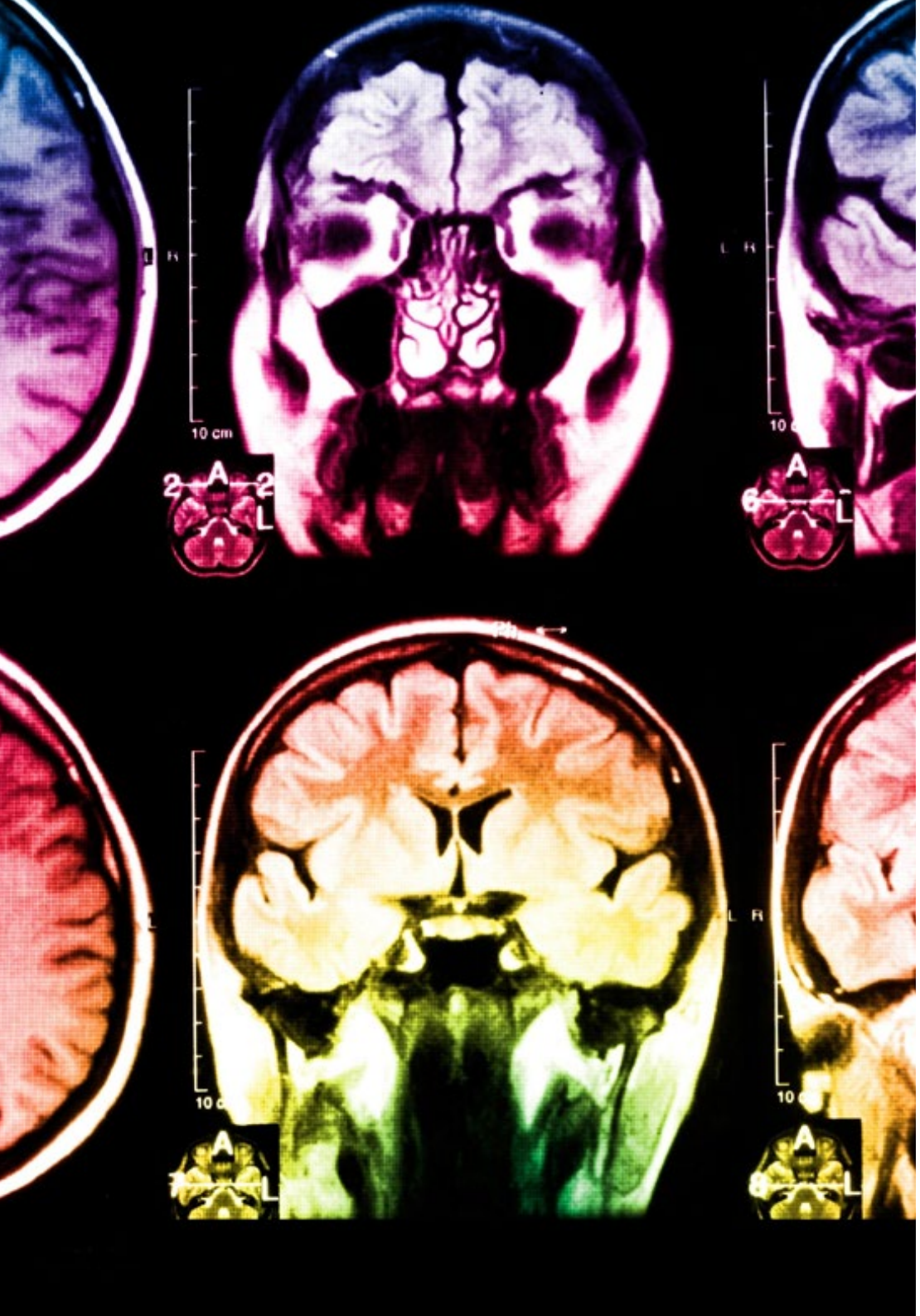
TECH coniuga efficacemente la metodologia del Caso di Studio con un sistema di apprendimento 100% online basato sulla ripetizione, che combina diversi elementi didattici in ogni lezione.

Potenziamo il Caso di Studio con il miglior metodo di insegnamento 100% online: il Relearning.

Nel 2019 abbiamo ottenuto i migliori risultati di apprendimento di tutte le università online del mondo.

In TECH imparerai con una metodologia all'avanguardia progettata per formare i manager del futuro. Questo metodo, all'avanguardia della pedagogia mondiale, si chiama Relearning.

La nostra università è l'unica autorizzata a utilizzare questo metodo di successo. Nel 2019, siamo riusciti a migliorare il livello di soddisfazione generale dei nostri studenti (qualità dell'insegnamento, qualità dei materiali, struttura del corso, obiettivi...) rispetto agli indicatori della migliore università online.



Nel nostro programma, l'apprendimento non è un processo lineare, ma avviene in una spirale (impariamo, disimpariamo, dimentichiamo e re-impariamo). Pertanto, combiniamo ciascuno di questi elementi in modo concentrico. Questa metodologia ha formato più di 650.000 laureati con un successo senza precedenti in campi diversi come la biochimica, la genetica, la chirurgia, il diritto internazionale, le competenze manageriali, le scienze sportive, la filosofia, il diritto, l'ingegneria, il giornalismo, la storia, i mercati e gli strumenti finanziari. Tutto questo in un ambiente molto esigente, con un corpo di studenti universitari con un alto profilo socio-economico e un'età media di 43,5 anni.

Il Relearning ti permetterà di apprendere con meno sforzo e più performance, impegnandoti maggiormente nella tua specializzazione, sviluppando uno spirito critico, difendendo gli argomenti e contrastando le opinioni: un'equazione diretta al successo.

Dalle ultime evidenze scientifiche nel campo delle neuroscienze, non solo sappiamo come organizzare le informazioni, le idee, le immagini e i ricordi, ma sappiamo che il luogo e il contesto in cui abbiamo imparato qualcosa è fondamentale per la nostra capacità di ricordarlo e immagazzinarlo nell'ippocampo, per conservarlo nella nostra memoria a lungo termine.

In questo modo, e in quello che si chiama Neurocognitive Context-dependent E-learning, i diversi elementi del nostro programma sono collegati al contesto in cui il partecipante sviluppa la sua pratica professionale.

Questo programma offre i migliori materiali didattici, preparati appositamente per i professionisti:

Materiale di studio

Tutti i contenuti didattici sono creati appositamente per il corso dagli specialisti che lo impartiranno, per fare in modo che lo sviluppo didattico sia davvero specifico e concreto.

Questi contenuti sono poi applicati al formato audiovisivo che supporterà la modalità di lavoro online di TECH. Tutto questo, con le ultime tecniche che offrono componenti di alta qualità in ognuno dei materiali che vengono messi a disposizione dello studente.

Master class

Esistono evidenze scientifiche sull'utilità dell'osservazione di esperti terzi.

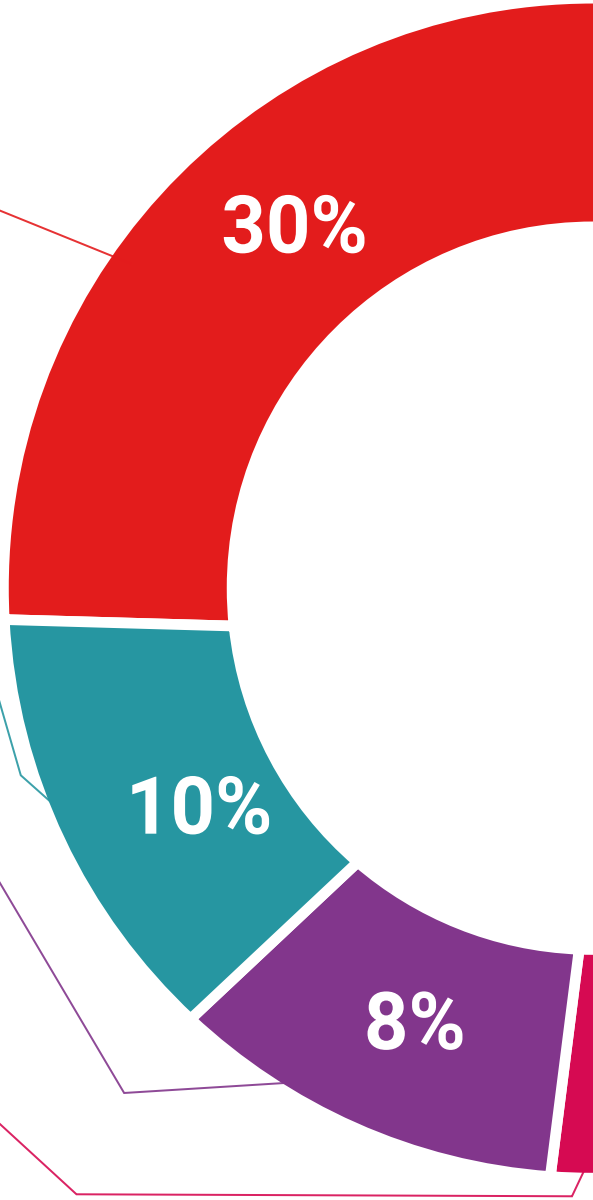
Imparare da un esperto rafforza la conoscenza e la memoria, costruisce la fiducia nelle nostre future decisioni difficili.

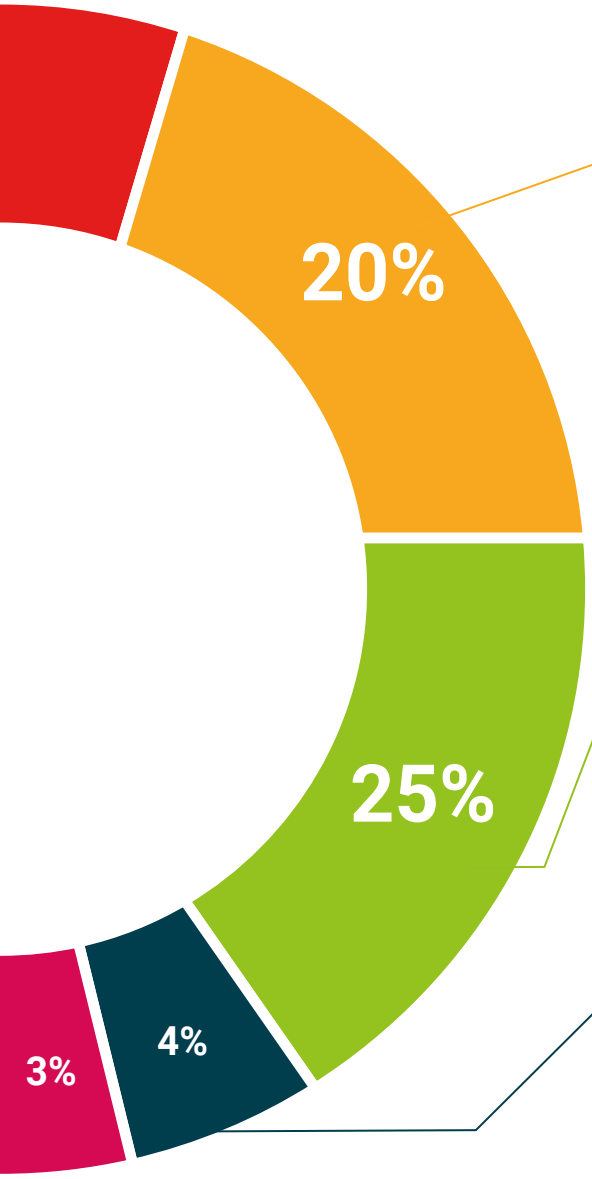
Pratiche di competenze e competenze

Svolgerai attività per sviluppare competenze e capacità specifiche in ogni area tematica. Pratiche e dinamiche per acquisire e sviluppare le competenze e le abilità che uno specialista deve sviluppare nel quadro della globalizzazione in cui viviamo.

Letture complementari

Articoli recenti, documenti di consenso e linee guida internazionali, tra gli altri. Nella biblioteca virtuale di TECH potrai accedere a tutto il materiale necessario per completare la tua specializzazione.





Casi di Studio

Completerai una selezione dei migliori casi di studio scelti appositamente per questo corso. Casi presentati, analizzati e monitorati dai migliori specialisti del panorama internazionale.

Riepiloghi interattivi

Il team di TECH presenta i contenuti in modo accattivante e dinamico in pillole multimediali che includono audio, video, immagini, diagrammi e mappe concettuali per consolidare la conoscenza.

Questo esclusivo sistema di specializzazione per la presentazione di contenuti multimediali è stato premiato da Microsoft come "Caso di successo in Europa".

Testing & Retesting

Valutiamo e rivalutiamo periodicamente le tue conoscenze durante tutto il programma con attività ed esercizi di valutazione e autovalutazione, affinché tu possa verificare come raggiungi progressivamente i tuoi obiettivi.

06

Titolo

Il Master Privato in Sviluppo di Software ti garantisce, oltre alla preparazione più rigorosa e aggiornata, l'accesso a una qualifica di Master Privato rilasciata da TECH Università Tecnologica.

Porta a termine questo programma e ricevi la tua qualifica universitaria senza spostamenti o fastidiose formalità”

Questo **Master Privato in Sviluppo di Software** possiede il programma più completo e aggiornato del mercato.

Dopo aver superato la valutazione, lo studente riceverà mediante lettera certificata* con ricevuta di ritorno, la sua corrispondente qualifica di **Master Privato** rilasciata da **TECH Università Tecnologica.**

Il titolo rilasciato da **TECH Università Tecnologica** esprime la qualifica ottenuta nel Master Privato, e riunisce tutti i requisiti comunemente richiesti da borse di lavoro, concorsi e commissioni di valutazione di carriere professionali.

Titolo: **Master Privato in Sviluppo di Software**

N. Ore Ufficiali: **1.500**


tech università tecnologica

Conferisce il presente

DIPLOMA

a

Dott./Dott.ssa _____, con documento d'identità n°_____

Per aver completato con esito positivo e accreditato il programma di

MASTER PRIVATO

in

Sviluppo di Software

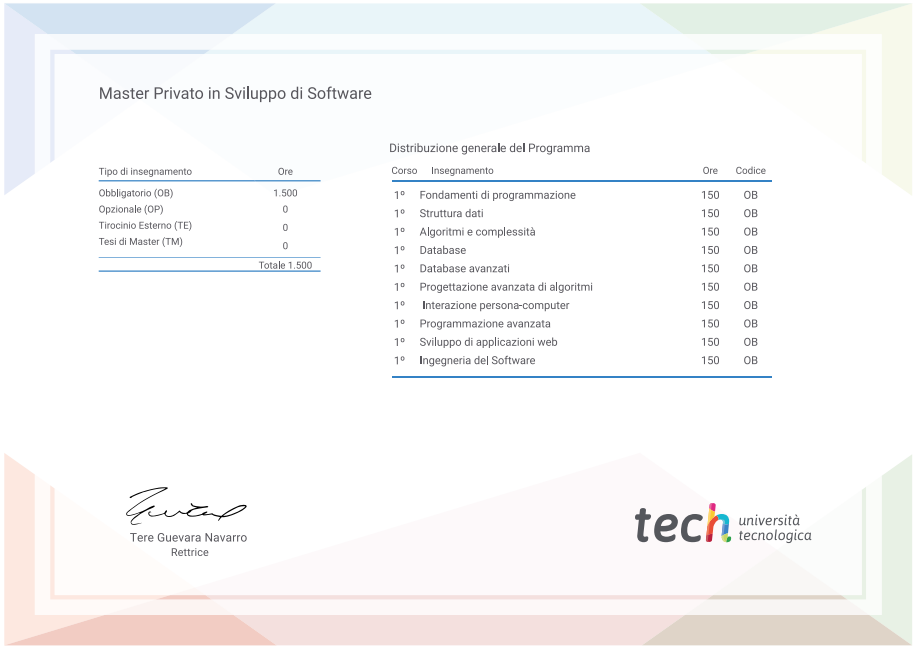
Si tratta di un titolo rilasciato da questa Università ed equivalente a 1.500 ore, con data di inizio gg/mm/aaaa e con data di fine gg/mm/aaaa.

TECH è un Istituto Privato di Istruzione Superiore riconosciuto dal Ministero della Pubblica Istruzione a partire dal 28 giugno 2018.

In data 17 Giugno 2020

Tere Guevara Navarro
Rettrice

codice unico TECH: AFWORD23S techtitute.com/titulos


Master Privato in Sviluppo di Software

Tipo di insegnamento	Ore
Obbligatorio (OB)	1.500
Opzionale (OP)	0
Tirocinio Esterno (TE)	0
Tesi di Master (TM)	0
	Totale 1.500

Distribuzione generale del Programma

Corso	Insegnamento	Ore	Codice
1°	Fondamenti di programmazione	150	OB
1°	Struttura dati	150	OB
1°	Algoritmi e complessità	150	OB
1°	Database	150	OB
1°	Database avanzati	150	OB
1°	Progettazione avanzata di algoritmi	150	OB
1°	Interazione persona-computer	150	OB
1°	Programmazione avanzata	150	OB
1°	Sviluppo di applicazioni web	150	OB
1°	Ingegneria del Software	150	OB

Tere Guevara Navarro
Rettrice

tech università tecnologica

*Se lo studente dovesse richiedere che il suo diploma cartaceo sia provvisto di Apostille dell'Aia, TECH EDUCATION effettuerà le gestioni opportune per ottenerla pagando un costo aggiuntivo

tech università tecnologica

Master Privato

Sviluppo di Software

- Modalità: **online**
- Durata: **12 mesi**
- Titolo: **TECH Università Tecnologica**
- Dedizione: **16 ore/settimana**
- Orario: **a scelta**
- Esami: **online**

Master Privato
Sviluppo di Software

tech università tecnologica