Professional Master's Degree
Software Development

tech technological university

tech technological university

Professional Master's Degree
Software Development

- Modality: **online**
- Duration: **12 months**
- Certificate: **TECH Technological University**
- Dedication: **16h/week**
- Schedule: **at your own pace**
- Exams: **online**

Website: www.techtitute.com/pk/information-technology/professional-master-degree/master-software-development

Index

01

Introduction

To participate in one of the areas with the greatest projection in the IT sector, professionals must be prepared scientifically and technologically, as well as to be able to efficiently face the challenges that arise in the professional practice of software engineering. This Professional Master's Degree is aimed at achieving a high mastery of Software Development, through the latest advances and developments in this field, by means of a study methodology of maximum impact and extraordinary flexibility.

Acquire the most comprehensive knowledge in software engineering, in the most up-to-date program of the online educational market and start working on developments in this dynamic professional field"

With the advance of new technologies, software has become an extremely important element nowadays. In recent years, the need to be able to develop software products with the appropriate functionality and quality, while respecting the established time and budget, has become evident.

This program is aimed at those people interested in reaching a higher level of knowledge about Software Development. The main objective is to enable students to apply the knowledge acquired in this Professional Master's Degree in the real world, in a work environment that reproduces the conditions that may be encountered in the future, in a rigorous and realistic manner.

Take advantage of the opportunity to take this educational program in a 100% online format, without having to give up obligations, and making it easy to continue studying. Update your knowledge and obtain a Professional Master's Degree to continue growing personally and professionally.

You will gain extensive knowledge in the field of software engineering, but also in the field of computating and computer structure, including the mathematical, statistical and physical principles that are essential in engineering.

Take advantage of the opportunity to take this educational program in a 100% online format, without having to give up obligations, and making it easy to continue studying. Update your knowledge and get your Professional Master's Degree to continue growing personally and professionally.

This **Professional Master's Degree in Software Development** contains the most complete and up-to-date program on the market. The most important features include:

- Development of 100 simulated scenarios presented by experts in Software Development.
- Its graphic, schematic and eminently practical contents, with which they are conceived, gather scientific and practical information on Software Development
- News on the latest developments in Software Development
- It contains practical exercises where the self-evaluation process can be carried out to improve learning
- Interactive learning system based on the case method and its application to real practice
- All of this will be complemented by theoretical lessons, questions to the expert, debate forums on controversial topics, and individual reflection assignments
- Content that is accessible from any fixed or portable device with an Internet connection

This program will allow you to learn about the basic structure of a computer and its software, as a basis for increasing your skills”

Learn everything you need to work with programming languages safely, incorporating to your knowledge the interpretation and design of basic algorithms to work in programming"

An educational program that will enable you to understand how to operate and intervene on all the essential elements of a computer program.

Get to know the latest data systems on the market, learn how to design advanced algorithms and all the aspects that a highly competent professional must master.

Its teaching staff includes professionals belonging to the world of Software Development, who bring to this program the experience of their work, as well as recognized specialists belonging to reference companies and prestigious universities.

Thanks to its multimedia content developed with the latest educational technology, they will allow the professional a situated and contextual learning, that is to say, a simulated environment that will provide an immersive learning programmed to train in real situations.

This program is designed around Problem-Based Learning, whereby the professional must try to solve the different professional practice situations that arise throughout the program. To do so, the professional will be assisted by an innovative interactive video system created by recognized experts in Software Development with extensive teaching experience.

02

Objectives

The objective of this program is to provide professionals working in Software Development with the knowledge and skills necessary to carry out their activity using the most advanced protocols and techniques of the moment. Through a work approach that is totally adaptable to the student, this Professional Master's Degree will progressively lead you to acquire the skills that will propel you to a higher professional level.

You will delve into the field of computating and computer structure; essential subjects for any software developer"

General Objectives

- Prepare scientifically and technologically, as well as to develop the professional practice of software engineering, with a transversal and versatile approach adapted to the new technologies and innovations in this field
- Obtain extensive knowledge in the field of software engineering, but also in the field of computation and computer structure, including the mathematical, statistical and physical basis essential in engineering

Achieve the level of knowledge you desire and master Software Development with this high-level training"

Specific Objectives

Module 1. Programming Fundamentals

- Understand the basic structure of a computer, software and general purpose programming languages
- Learn to design and interpret algorithms, which are the necessary basis for developing computer programs
- Understand the essential elements of a computer program, such as the different types of data, operators, expressions, statements, I/O and control statements
- Understand the different data structures available in general purpose programming languages, both static and dynamic, as well as to acquire the essential knowledge for file handling
- Know the different testing techniques in computer programs and the importance of generating good documentation together with good source code
- Learn the basic concepts of the C++ programming language, one of the most widely used languages in the world

Module 2. Data Structure

- Learn the fundamentals of C++ programming, including classes, variables, conditional expressions and objects
- Understand abstract data types, linear data structure types, simple and complex hierarchical data structures, as well as their implementation in C++
- Understand the operation of advanced data structures other than the usual ones
- Know the theory and practice related to the use of priority heaps and queues
- Learn how Hash tables work as abstract data types and functions
- Understand Graph theory, as well as advanced Graph algorithms and concepts

Module 3. Algorithm and Complexity

- Learn the main strategies for algorithm design, as well as the different methods and measures for algorithm computation
- Know the main sorting algorithms used in software development
- Understand the operation of the different algorithms with trees, Heaps and Graphs
- Understand the operation of Greedy algorithms, their strategy and examples of their use in the main known problems. We will also learn the use of greedy algorithms on graphs
- We will learn the main strategies of minimum path search, with the approach of essential problems of the field and algorithms for their resolution
- Understand the Backtracking technique and its main uses, as well as other alternative techniques

Module 4. Databases

- Learn the different applications and purposes of database systems, as well as their operation and architecture
- Understand the relational model, from its structure and operations to extended relational algebra
- Learn in depth what SQL databases are, how they work, the definition of data and the creation of queries from the most basic to the most advanced and complex
- Learn how to design databases using the entity-relationship model, how to create diagrams and the characteristics of the extended E-R model
- Delve into the design of relational databases, analyzing the different normal forms and decomposition algorithms
- Lay the foundations for understanding the operation of NoSQL databases and to introduce the MongoDB database

Module 5. Advanced Databases

- Introduce the different database systems currently available on the market
- Learn the use of XML and databases for the web
- Understand the operation of advanced databases such as parallel and distributed databases
- Understand the importance of indexing and association in database systems
- Understand how transactional processing and retrieval systems work
- Acquire knowledge related to non-relational databases and data mining

Module 6. Advanced Algorithm Design

- Delve into advanced algorithm design, analyzing recursive and divide-and-conquer algorithms, as well as performing amortized analysis
- Understand the concepts of dynamic programming and algorithms for NP problems.
- Understand the operation of combinatorial optimization, as well as the different randomization algorithms and parallel algorithms
- Know and understand the operation of the different local and candidate search methods
- Learn the mechanisms of formal verification of programs and iterative programs, including first-order logic and Hoare's formal system
- Learn the operation of some of the main numerical methods such as the bisection method, the Newton Raphson method and the secant method

Module 7. Human-Computer Interaction

- Acquire solid knowledge related to human-computer interaction and the creation of usable interfaces
- Understand the importance of application usability and why it is important to take it into account when designing our software
- Understand the different types of human diversity, the limitations they imply and how to adapt interfaces according to the specific needs of each of them
- Learn the process of interface design, from requirements analysis to evaluation, going through the different intermediate stages necessary to carry out an adequate interface
- Know the different accessibility guidelines, the standards that establish them and the tools that allow us to assess them
- Understand the different methods of interaction with the computer, by means of peripherals and devices

Module 8. Advanced Programming

- In-depth knowledge of programming, especially as it relates to object-oriented programming, and the different types of relationships between existing classes
- Know the different design patterns for object-oriented problems
- Learn about event-driven programming and the development of user interfaces with Qt
- Acquire the essential knowledge of Concurrent Programming, processes and threads
- Learn how to manage the use of threads and synchronization, as well as the resolution of common problems within Concurrent Programming
- Understand the importance of documentation and testing in software development

Module 9. Development of Web Applications

- Know the characteristics of the HTML markup language and its use in web creation together with CSS style sheets
- Learn to use the browser-oriented programming language JavaScript, and some of its main features
- Understand the concepts of Component Oriented Programming and Component Architecture
- Learn how to use the Bootstrap Frontend Framework for website design
- Understand the structure of the controller view model in the development of dynamic web sites
- Know the service-oriented architecture and the basics of the HTTP protocol

Module 10. Software Engineering

- Lay the foundations of software engineering and modeling, learning the main processes and concepts
- Understand the software process and the different models for its development including agile technologies
- Understand requirements engineering, their development, elaboration, negotiation and validation
- Learn the modeling of requirements and the different elements such as scenarios, information, analysis classes, flow, behavior and patterns
- Understand the concepts and processes of software design, learning also about architecture design and design at component level and based on patterns
- Know the main standards related to software quality and project management

03

Skills

By passing the assessments of the Professional Master's Degree in Software Development, you will have acquired the professional skills necessary to carry out quality work and you will also be able to acquire new skills and techniques that will help you to complement the computer knowledge you previously possessed.

Enhance your skills in Software Development and move to the next level as a professional in this constantly evolving field"

General Skill

» Respond to the current needs of the field of Software Development

An exceptional program in terms of its density, its current relevance and the way in which it is offered, which will allow you to advance quickly and efficiently"

Specific Skills

- Be able to understand the basic structure of a computer, software and general purpose programming languages
- Know how to apply the fundamentals of C++ programming, including classes, variables, conditional expressions and objects
- Know, in depth, the main strategies for algorithm design, as well as the different methods and measures for their calculation
- Know the different applications and purposes of database systems, as well as their operation and architecture, and to apply them on a day-to-day basis
- Be able to introduce the different database systems currently on the market
- Know how to analyze recursive and divide and conquer algorithms, as well as how to perform amortized analysis
- Use the knowledge of human-computer interaction and the creation of usable interfaces in the daily practice of the profession
- Acquire in-depth knowledge of Programming
- Know the characteristics of the HTML markup language and its use in web creation together with CSS style sheets
- Be able to apply the main processes and concepts of the basics of software engineering and modeling

04

Structure and Content

The structure of the contents has been designed by a team of Computer Engineering professionals with the aim of ensuring that students of this Professional Master's Degree can learn efficiently and quickly. For this purpose, the contents have been organized in such a way that learning is intensive and constant, trying to maintain motivation based on the student's sense of progress.

An educational program aimed at achieving complete software development skills, which will propel you to a new professional level"

Module 1. Programming Fundamentals

1.1. Introduction to Programming
- 1.1.1. Basic Structure of a Computer
- 1.1.2. Software
- 1.1.3. Programming Languages
- 1.1.4. Life Cycle of a Software Application

1.2. Algorithm Design
- 1.2.1. Problem Solving
- 1.2.2. Descriptive Techniques
- 1.2.3. Algorithm Elements and Structure

1.3. Elements of a Program
- 1.3.1. C++ Origin and Features
- 1.3.2. Development Environment
- 1.3.3. Concept of Program
- 1.3.4. Types of Fundamental Data
- 1.3.5. Operators
- 1.3.6. Expressions
- 1.3.7. Statements
- 1.3.8. Data Input and Output

1.4. Control Sentences
- 1.4.1. Statements
- 1.4.2. Branches
- 1.4.3. Loops

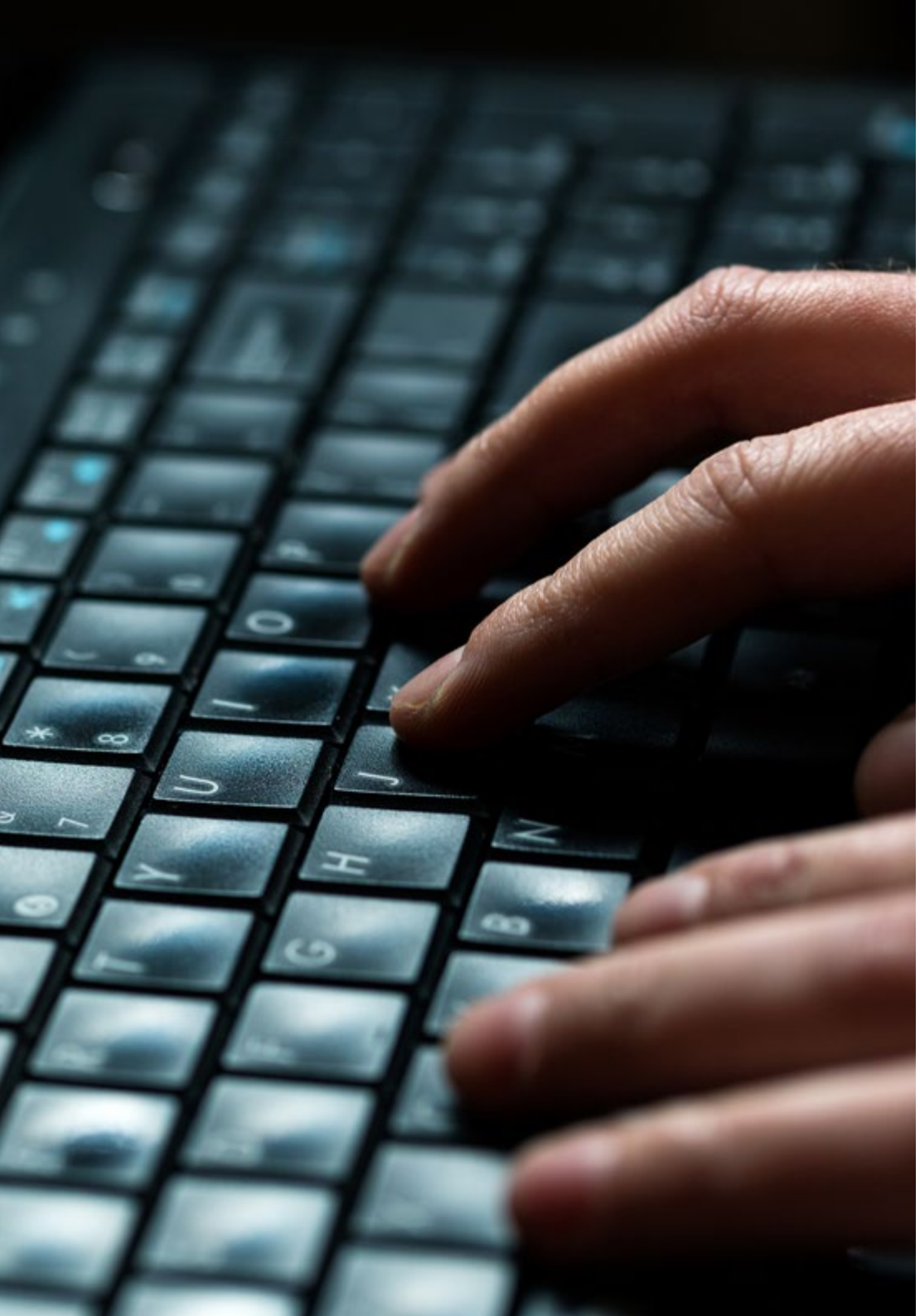
1.5. Abstraction and Modularity: Functions
- 1.5.1. Modular Design
- 1.5.2. Concept of Function and Utility
- 1.5.3. Definition of a Function
- 1.5.4. Execution Flow in a Function Call
- 1.5.5. Function Prototypes
- 1.5.6. Results Return
- 1.5.7. Calling a Function: Parameters
- 1.5.8. Passing Parameters by Reference and by Value
- 1.5.9. Scope Identifier

1.6. Static Data Structures
- 1.6.1. Arrays
- 1.6.2. Matrices. Polyhedra
- 1.6.3. Searching and Sorting
- 1.6.4. Chaining: I/O Functions for Chains
- 1.6.5. Structures. Unions
- 1.6.6. New Types of Data

1.7. Dynamic Data Structures: Pointers
- 1.7.1. Concept. Definition of Pointer
- 1.7.2. Pointer Operators and Operations
- 1.7.3. Pointer Arrays
- 1.7.4. Pointers and Arrays
- 1.7.5. Chain Pointers
- 1.7.6. Structure Pointers
- 1.7.7. Multiple Indirection
- 1.7.8. Function Pointers
- 1.7.9. Function, Structure and Array Passing as Function Parameters

1.8. Files
- 1.8.1. Basic Concepts
- 1.8.2. File Operations
- 1.8.3. Types of Files
- 1.8.4. File Organization
- 1.8.5. Introduction to C++ Files
- 1.8.6. Managing Files

1.9. Recursion
- 1.9.1. Definition of Recursion
- 1.9.2. Types of Recursion
- 1.9.3. Advantages and Disadvantages
- 1.9.4. Considerations
- 1.9.5. Iterative Recursive Conversion
- 1.9.6. Recursion Stack



1.10. Testing and Documentation
- 1.10.1. Program Testing
- 1.10.2. White Box Testing
- 1.10.3. Black Box Testing
- 1.10.4. Testing Tools
- 1.10.5. Program Documentation

Module 2. Data Structure

2. 1. Introduction to C ++ Programming
- 2.1.1. Classes, Constructors, Methods and Attributes
- 2.1.2. Variables
- 2.1.3. Conditional Expressions and Loops
- 2.1.4. Objects

2.2. Abstract Data Types (ADT)
- 2.2.1. Types of Data
- 2.2.2. Basic Structures and TADs
- 2.2.3. Vectors and Arrays

2.3. Linear data Structures
- 2.3.1. TAD List Definition
- 2.3.2. Linked and Doubly Linked Lists
- 2.3.3. Sorted Lists
- 2.3.4. Lists in C++
- 2.3.5. TAD Stack
- 2.3.6. TAD Queue
- 2.3.7. Stack and Queue in C++

2.4. Hierarchical Data Structures
- 2.4.1. TAD Tree
- 2.4.2. Paths
- 2.4.3. N-Ary Trees
- 2.4.4. Binary Trees
- 2.4.5. Binary Search Trees

2.5. Hierarchical Data Structures: Complex Trees
  2.5.1. Perfectly Balanced or Minimum Height Trees
  2.5.2. Multipath Trees
  2.5.3. Bibliographical References
2.6. Mounds and Priority Queue
  2.6.1. TAD Mounds
  2.6.2. TAD Priority Queue
2.7. Hash Tables
  2.7.1. TAD Hash Table
  2.7.2. Hash Functions
  2.7.3. Hash Function in Hash Tables
  2.7.4. Redispersion
  2.7.5. Open Hash Tables
2.8. Graphs
  2.8.1. TAD Graph
  2.8.2. Graph Types
  2.8.3. Graphical Representation and Basic Operations
  2.8.4. Graph Design
2.9. Advanced Graph Algorithms and Concepts
  2.9.1. Graph Problems
  2.9.2. Path Algorithms
  2.9.3. Search or Path Algorithms
  2.9.4. Other Algorithms
2.10. Other Data Structures
  2.10.1. Sets
  2.10.2. Parallel Arrays
  2.10.3. Symbol Tables
  2.10.4. Tries

Module 3. Algorithm and Complexity

3.1. Introduction to Algorithm Design Strategies
  3.1.1. Recursion
  3.1.2. Divide and Conquer
  3.1.3. Other Strategies
3.2. Efficiency and Analysis of Algorithms
  3.2.1. Efficiency Measures
  3.2.2. Measuring the Size of the Input
  3.2.3. Measuring Execution Time
  3.2.4. Worst, Best and Average Case
  3.2.5. Asymptotic Notation
  3.2.6. Criteria for Mathematical Analysis of Non-Recursive Algorithms
  3.2.7. Mathematical Analysis of Recursive Algorithms
  3.2.8. Empirical Analysis of Algorithms
3.3. Sorting Algorithms
  3.3.1. Concept of Sorting
  3.3.2. Bubble Sorting
  3.3.3. Sorting by Selection
  3.3.4. Sorting by Insertion
  3.3.5. Merge Sort
  3.3.6. Quick Sort
3.4. Algorithms with Trees
  3.4.1. Tree Concept
  3.4.2. Binary Trees
  3.4.3. Tree Paths
  3.4.4. Representing Expressions
  3.4.5. Ordered Binary Trees
  3.4.6. Balanced Binary Trees
3.5. Algorithms Using Heaps
  3.5.1. Heaps
  3.5.2. The Heapsort Algorithm
  3.5.3. Priority Queues

3.6. Graph Algorithms
- 3.6.1. Representation
- 3.6.2. Traversal in Width
- 3.6.3. Depth Travel
- 3.6.4. Topological Sorting

3.7. Greedy Algorithms
- 3.7.1. Greedy Strategy
- 3.7.2. Elements of the Greedy Strategy
- 3.7.3. Currency Exchange
- 3.7.4. Traveler's Problem
- 3.7.5. Backpack Problem

3.8. Minimal Path Finding
- 3.8.1. The Minimum Path Problem
- 3.8.2. Negative Arcs and Cycles
- 3.8.3. Dijkstra's Algorithm

3.9. Greedy Algorithms on Graphs
- 3.9.1. The Minimum Covering Tree
- 3.9.2. Prim's Algorithm
- 3.9.3. Kruskal's Algorithm
- 3.9.4. Complexity Analysis

3.10. Backtracking
- 3.10.1. Backtracking
- 3.10.2. Alternative Techniques

Module 4. Databases

4.1. Applications and Purposes of Database Systems
- 4.1.1. Applications of the Different Database Systems
- 4.1.2. Purpose of the Different Database Systems
- 4.1.3. View of the Data

4.2. Database and Architecture
- 4.2.1. Relational Database
- 4.2.2. Database Design
- 4.2.3. Object-Based and Semi-Structured Databases
- 4.2.4. Data Storage and Queries
- 4.2.5. Transaction Management
- 4.2.6. Data Mining and Analysis
- 4.2.7. Database Architecture

4.3. The Relational Model: Structure, Operations and Extended Relational Algebra
- 4.3.1. The Structure of Relational Databases
- 4.3.2. Fundamental Operations in the Relational Algebra
- 4.3.3. Other Relational Algebra Operations
- 4.3.4. Extended Relational Algebra Operations
- 4.3.5. Null Values
- 4.3.6. Database Modification

4.4. SQL (I)
- 4.4.1. What is SQL?
- 4.4.2. The Definition of Data
- 4.4.3. Basic Structure of SQL Queries
- 4.4.4. Operations on Sets
- 4.4.5. Aggregation Functions
- 4.4.6. Null Values

4.5. SQL (II)
- 4.5.1. Nested Subqueries
- 4.5.2. Complex Queries
- 4.5.3. Views
- 4.5.4. Cursors
- 4.5.5. Complex Queries
- 4.5.6. Triggers

4.6. Database Design and the E-R Model
- 4.6.1. Overview of the Design Process
- 4.6.2. The Entity-Relationship Model
- 4.6.3. Restrictions

4.7. Entity-Relationship Diagrams
- 4.7.1. Entity-Relationship Diagrams
- 4.7.2. Aspects of Entity-Relationship Design
- 4.7.3. Weak Entity Sets

4.8. The Extended Entity-Relationship Model
- 4.8.1. Characteristics of the Extended E-R Model
- 4.8.2. Design of a Database
- 4.8.3. Reduction to Relational Schemas

4.9. Designing from Relational Databases
- 4.9.1. Characteristics of Good Relational Designs
- 4.9.2. Atomic Domains and the First Normal Form (1FN)
- 4.9.3. Decomposition by Functional Dependencies
- 4.9.4. Theory of Functional Dependencies
- 4.9.5. Decomposition Algorithms
- 4.9.6. Decomposition by Means of Multivalued Dependencies
- 4.9.7. More Normal Forms
- 4.9.8. Database Design Process

4.10. NoSQL Databases
- 4.10.1. What are NoSQL Databases?
- 4.10.2. Analysis of the Different NoSQL Options and their Characteristics.
- 4.10.3. MongoDB

Module 5. Advanced Databases

5.1. Introduction to the Different Database Systems
- 5.1.1. Historical Recap
- 5.1.2. Hierarchical Databases
- 5.1.3. Network Databases
- 5.1.4. Relational Databases
- 5.1.5. Non-Relational Databases

5.2. XML and Databases for the Web
- 5.2.1. Validation of XML Documents
- 5.2.2. XML Document Transformations
- 5.2.3. XML Data Storage
- 5.2.4. XML Relational Databases
- 5.2.5. SQL/XML
- 5.2.6. Native XML Databases

5.3. Parallel Databases
- 5.3.1. Parallel Systems
- 5.3.2. Parallel Database Architectures
- 5.3.4. Parallelism in Queries
- 5.3.5. Query Parallelism
- 5.3.6. Design of Parallel Systems
- 5.3.7. Parallel Processing in SQL

5.4. Distributed Databases
- 5.4.1. Distributed Systems
- 5.4.2. Distributed Storage
- 5.4.3. Availability
- 5.4.4. Distributed Query Processing
- 5.4.5. Distributed Database Providers

5.5. Indexing and Association
- 5.5.1. Ordered Indexes
- 5.5.2. Dense and Sparse Indexes
- 5.5.3. Multilevel Indices
- 5.5.4. Index Updating
- 5.5.5. Static Association
- 5.5.6. How to Use Indexes in Databases

5.6. Introduction to Transactional Processing
- 5.6.1. States of a Transaction
- 5.6.2. Implementation of atomicity and durability.
- 5.6.3. Sequentiality
- 5.6.4. Recoverability
- 5.6.5. Isolation Implementation

5.7. Recovery Systems
- 5.7.1. Failure Classification
- 5.7.2. Storage Structures
- 5.7.3. Recovery and Atomicity
- 5.7.4. Retrieval Based on Historical Record
- 5.7.5. Concurrent Transactions and Retrieval
- 5.7.6. High Availability in Databases

5.8. Execution and Processing of Queries
- 5.8.1. Cost of a Query
- 5.8.2. Selection Operation
- 5.8.3. Sorting
- 5.8.4. Introduction to Query Optimization
- 5.8.5. Performance Monitoring

5.9. Non-Relational Databases
- 5.9.1. Document-Oriented Databases
- 5.9.2. Graph Oriented Databases
- 5.9.3. Key-Value Databases

5.10. Data Warehouse, OLAP and Data Mining
- 5.10.1. Components of Data Warehouses
- 5.10.2. Architecture of a Data Warehouse
- 5.10.3. OLAP
- 5.10.4. Data Mining Functionality
- 5.10.5. Other Types of Mining

Module 6. Advanced Algorithm Design

6.1. Analysis of Recursive and Divide and Conquer Algorithms
- 6.1.1. Posing and Solving Homogeneous and Non-Homogeneous Recurrence Equations
- 6.1.2. General Description of the Divide and Conquer Strategy

6.2. Amortized Analysis
- 6.2.1. Aggregate Analysis
- 6.2.2. The Accounting Method
- 6.2.3. The Potential Method

6.3. Dynamic Programming and Algorithms for NP Problems
- 6.3.1. Characteristics of Dynamic Programming
- 6.3.2. Backtracking
- 6.3.3. Branching and Pruning

6.4. Combinatorial Optimization
- 6.4.1. Representation
- 6.4.2. 1D Optimization

6.5. Randomization Algorithms
- 6.5.1. Examples of Randomization Algorithms
- 6.5.2. The Buffon Theorem
- 6.5.3. Monte Carlo Algorithm
- 6.5.4. Las Vegas Algorithm

6.6. Local and Candidate Search
- 6.6.1. Gradient Ascent
- 6.6.2. Hill Climbing
- 6.6.3. Simulated Annealing
- 6.6.4. Tabu Search
- 6.6.5. Candidate Search

6.7. Formal Verification of Programs
- 6.7.1. Specification of Functional Abstractions
- 6.7.2. The Language of First-Order Logic
- 6.7.3. Hoare's Formal System

6.8. Verification of Iterative Programs
- 6.8.1. Rules of Hoare's Formal System
- 6.8.2. Concept of Invariant Iterations

6.9. Numeric Methods
- 6.9.1. The Bisection Method
- 6.9.2. Newton Raphson's Method
- 6.9.3. The Secant Method

6.10. Parallel Algorithms
- 6.10.1. Parallel Binary Operations
- 6.10.2. Parallel Operations with Networks
- 6.10.3. Parallelism in Divide and Conquer
- 6.10.4. Parallelism in Dynamic Programming

Module 7. Human-Computer Interaction

7.1. Introduction to Human-Computer Interaction
- 7.1.1. What is Human-Computer Interaction
- 7.1.2. Relationship of Human-Computer Interaction with Other Disciplines
- 7.1.3. The User Interface
- 7.1.4. Usability and Accessibility
- 7.1.5. User Experience and User-Centered Design

7.2. The Computer and Interaction: User Interface and Interaction Paradigms
- 7.2.1. Interaction
- 7.2.2. Paradigms and Styles of Interaction
- 7.2.3. Evolution of User Interfaces
- 7.2.4. Classic User Interfaces: WIMP/GUI, Commands, Voice, Virtual Reality
- 7.2.5. Innovative User Interfaces: Mobile, Wearable, Collaborative, BCI

7.3. The Human Factor: Psychological and Cognitive Aspects
- 7.3.1. The Importance of the Human Factor in Interaction
- 7.3.2. Human Information Processing
- 7.3.3. The Input and Output of Information: Visual, Auditory, and Tactile
- 7.3.4. Perception and Attention
- 7.3.5. Knowledge and Mental Models: Representation, Organization, and Acquisition

7.4. The Human Factor: Sensory and Physical Limitations
- 7.4.1. Functional Diversity, Disability and Impairment
- 7.4.2. Visual Diversity
- 7.4.3. Hearing Diversity
- 7.4.4. Cognitive Diversity
- 7.4.5. Motor Diversity
- 7.4.6. The Case of Digital Immigrants

7. 5. The Design Process (I): Requirements Analysis for User Interface Design
- 7.5.1. User-Centered Design
- 7.5.2. What is Requirements Analysis?
- 7.5.3. Information Gathering
- 7.5.4. Analysis and Interpretation of the Information
- 7.5.5. Usability and Accessibility Analysis

7.6. The Design Process (II): Prototyping and Task Analysis
- 7.6.1. Conceptual Design
- 7.6.2. Prototyping
- 7.6.3. Hierarchical Task Analysis

7.7. The Design Process (III): Evaluation
- 7.7.1. Evaluation in the Design Process: Objectives and Methods
- 7.7.2. Evaluation Methods Without Users
- 7.7.3. Evaluation Methods with Users
- 7.7.4. Evaluation Standards and Norms

7.8. Accessibility: Definition and Guidelines
- 7.8.1. Accessibility and Universal Design
- 7.8.2. The WAI Initiative and the WCAG Guidelines
- 7.8.3. WCAG 2.0 and 2.1 Guidelines

7.9. Accessibility: Evaluation and Functional Diversity
- 7.9.1. Web Accessibility Evaluation Tools
- 7.9.2. Accessibility and Functional Diversity

7.10. The Computer and Interaction: Peripherals and Devices
- 7.10.1. Traditional Devices and Peripherals
- 7.10.2. Alternative Devices and Peripherals
- 7.10.3. Cell Phones and Tablets
- 7.10.4. Functional Diversity, Interaction and Peripherals

Module 8. Advanced Programming

8.1. Introduction to Object-Oriented Programming
- 8.1.1. Introduction to Object-Oriented Programming
- 8.1.2. Class Design
- 8.1.3. Introduction to UML for Problem Modeling

8.2. Relationships Between Classes
- 8.2.1. Abstraction and Inheritance
- 8.2.2. Advanced Inheritance Concepts
- 8.2.3. Polymorphism
- 8.2.4. Composition and Aggregation

8.3. Introduction to Design Patterns for Object-Oriented Problems
- 8.3.1. What are Design Patterns?
- 8.3.2. Factory Pattern
- 8.3.4. Singleton Pattern
- 8.3.5. Observer Pattern
- 8.3.6. Composite Pattern

8.4. Exceptions
- 8.4.1. What are Exceptions?
- 8.4.2. Exception Catching and Handling
- 8.4.3. Throwing Exceptions
- 8.4.4. Exception Creation

8.5. User Interfaces
- 8.5.1. Introduction to Qt
- 8.5.2. Positioning
- 8.5.3. What Are Events?
- 8.5.4. Events: Definition and Catching
- 8.5.5. User Interface Development

8.6. Introduction to Concurrent Programming
- 8.6.1. Introduction to Concurrent Programming
- 8.6.2. The Concept of Process and Thread
- 8.6.3. Interaction Between Processes or Threads
- 8.6.4. Threads in C++
- 8.6.6. Advantages and Disadvantages of Concurrent Programming

8.7. Thread Management and Synchronization
- 8.7.1. Life Cycle of a Thread
- 8.7.2. Thread Class
- 8.7.3. Thread Planning
- 8.7.4. Thread Groups
- 8.7.5. Daemon Threads
- 8.7.6. Synchronization
- 8.7.7. Locking Mechanisms
- 8.7.8. Communication Mechanisms
- 8.7.9. Monitors

8.8. Common Problems in Concurrent Programming
- 8.8.1. The Problem of Consuming Producers
- 8.8.2. The Problem of Readers and Writers
- 8.8.3. The Problem of the Philosophers' Dinner Party

8.9. Software Documentation and Testing
- 8.9.1. Why is it Important to Document Software?
- 8.9.2. Design Documentation
- 8.9.3. Documentation Tool Use

8.10. Software Testing
- 8.10.1. Introduction to Software Testing
- 8.10.2. Types of Tests
- 8.10.3. Unit Test
- 8.10.4. Integration Test
- 8.10.5. Validation Test
- 8.10.6. System Test

Module 9. Development of Web Applications

9.1. HTML5 Markup Languages
- 9.1.1. HTML Basics
- 9.1.2. New HTML 5 Elements
- 9.1.3. Forms: New Controls

9.2. Introduction to CSS Style Sheets
- 9.2.1. First Steps with CSS
- 9.2.2. Introduction to CSS3

9.3. Browser Scripting Language: JavaScript
- 9.3.1. JavaScript Basics
- 9.3.2. DOM
- 9.3.3. Events
- 9.3.4. JQuery
- 9.3.5. Ajax

9.4. Concept of Component-Oriented Programming
- 9.4.1. Context
- 9.4.2. Components and Interfaces
- 9.4.3. States of a Component

9.5. Component Architecture
- 9.5.1. Current Architectures
- 9.5.2. Component Integration and Deployment

9.6. Frontend Framework: Bootstrap
- 9.6.1. Grid Design
- 9.6.2. Forms
- 9.6.3. Components

9.7. Model View Controller
- 9.7.1. Web Development Methods
- 9.7.2. Design Pattern: MVC

9.8. Information Grid Technologies
- 9.8.1. Increased Computing Resources
- 9.8.2. Concept of Grid Technology

9.9. Service-Oriented Architecture
- 9.9.1. SOA and Web Services
- 9.9.2. Topology of a Web Service
- 9.9.3. Platforms for Web Services

9.10. HTTP Protocol
- 9.10.1. Messages
- 9.10.2. Persistent Sessions
- 9.10.3. Cryptographic System
- 9.10.4. HTTPS Protocol Operation

Module 10. Software Engineering

10.1. Introduction to Software Engineering and Modeling
- 10.1.1. The Nature of Software
- 10.1.2. The Unique Nature of WebApps
- 10.1.3. Software Engineering
- 10.1.4. The Software Process
- 10.1.5. Software Engineering Practice
- 10.1.6. Software Myths
- 10.1.7. How It All Begins
- 10.1.8. Object-Oriented Concepts
- 10.1.9. Introduction to UML

10.2. The Software Process
- 10.2.1. A General Process Model
- 10.2.2. Prescriptive Process Models
- 10.2.3. Specialized Process Models
- 10.2.4. The Unified Process
- 10.2.5. Personal and Team Process Models
- 10.2.6. What is Agility?
- 10.2.7. What is an Agile Process?
- 10.2.8. Scrum
- 10.2.9. Agile Process Toolkit

10.3. Principles Guiding Software Engineering Practice
- 10.3.1. Principles Guiding the Process
- 10.3.2. Principles Guiding the Practice
- 10.3.3. Principles of Communication
- 10.3.4. Planning Principles
- 10.3.5. Modeling Principles
- 10.3.6. Construction Principles
- 10.3.7. Deployment Principles

10.4. Understanding the Requirements
- 10.4.1. Requirements Engineering
- 10.4.2. Establish the Basis
- 10.4.3. Inquiry of Requirements
- 10.4.4. Development of Cases Studies
- 10.4.5. Elaboration of the Requirements Model
- 10.4.6. Negotiation of Requirements
- 10.4.7. Validation of Requirements

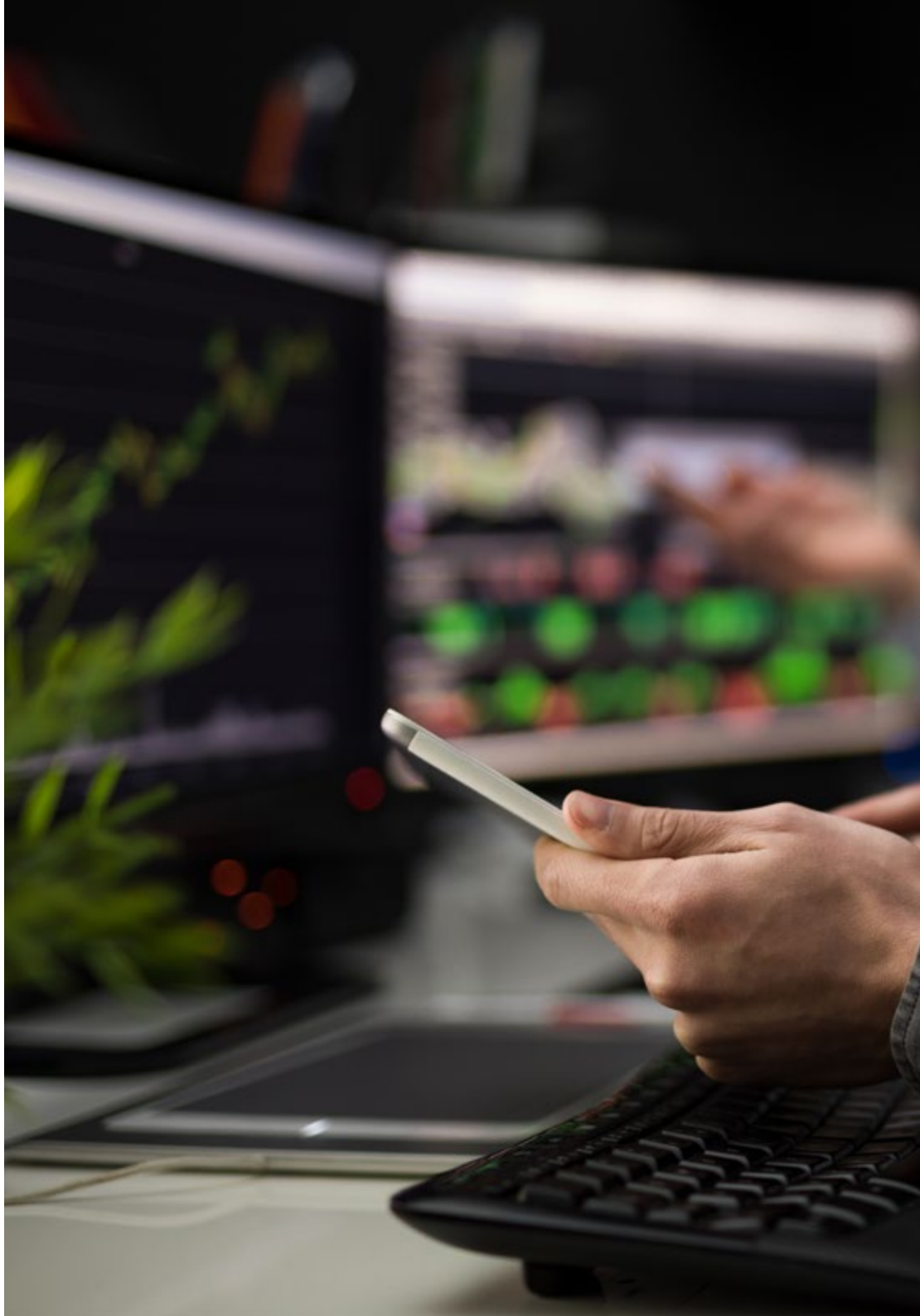
10.5. Requirements Modeling: Scenarios, Information and Analysis Classes
- 10.5.1. Analysis of Requirements
- 10.5.2. Scenario-Based Modeling
- 10.5.3. UML Models that provide the Case Study
- 10.5.4. Data Modeling Concepts
- 10.5.5. Class-Based Modeling
- 10.5.6. Class Diagrams

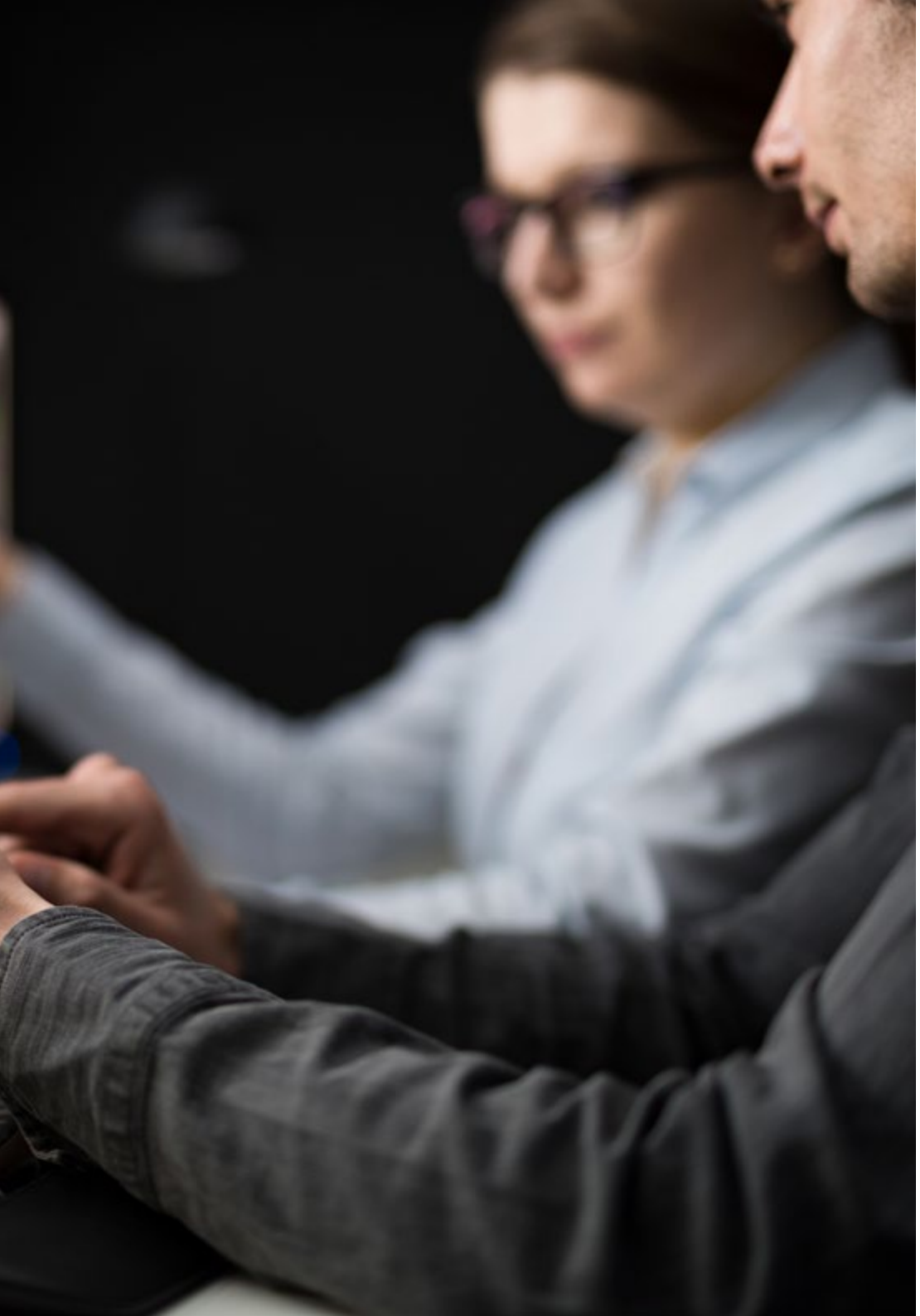
10.6. Requirements Modeling: Flow, Behavior and Patterns
- 10.6.1. Requirements that Shape Strategies
- 10.6.2. Flow-Oriented Modeling
- 10.6.3. Status Diagrams
- 10.6.4. Creation of a Behavioral Model
- 10.6.5. Sequence Diagrams
- 10.6.6. Communication Diagrams
- 10.6.7. Patterns for Requirements Modeling

10.7. Design Concepts
- 10.7.1. Design in the Software Engineering Context
- 10.7.2. The Design Process
- 10.7.3. Design Concepts
- 10.7.4. Object-Oriented Design Concepts
- 10.7.5. Model of the Design

10.8. Designing the Architecture:
- 10.8.1. Software Architecture
- 10.8.2. Architectural Genres
- 10.8.3. Architectural Styles
- 10.8.4. Architectural Design
- 10.8.5. Evolution of Alternative Designs for Architecture
- 10.8.6. Mapping the Architecture Using the Data Flow

10.9. Component-Level and Pattern-Based Design
- 10.9.1. What is a Component?
- 10.9.2. Class-Based Component Design
- 10.9.3. Realization of the Design at the Component Level
- 10.9.4. Design of Traditional Components
- 10.9.5. Component-Based Development
- 10.9.6. Design Patterns
- 10.9.7. Pattern-Based Software Design
- 10.9.8. Architectural Patterns
- 10.9.9. Design Patterns at the Component Level
- 10.9.10. User Interface Design Patterns





10.10. Software Quality and Project Management
- 10.10.1. Quality
- 10.10.2. Software Quality
- 10.10.3. The Software Quality Dilemma
- 10.10.4. Achieving Software Quality
- 10.10.5. Software Quality Assurance
- 10.10.6. The Administrative Spectrum
- 10.10.7. The Staff
- 10.10.8. The product
- 10.10.9. The Process
- 10.10.10. The Project
- 10.10.11. Principles and Practices

A unique, key, and decisive educational experience to boost your professional development"

05

Methodology

This academic program offers students a different way of learning. Our methodology uses a cyclical learning approach: **Relearning**.

This teaching system is used, for example, in the most prestigious medical schools in the world, and major publications such as the **New England Journal of Medicine** have considered it to be one of the most effective.

Discover Relearning, a system that abandons conventional linear learning, to take you through cyclical teaching systems: a way of learning that has proven to be extremely effective, especially in subjects that require memorization"

Case Study to contextualize all content

Our program offers a revolutionary approach to developing skills and knowledge. Our goal is to strengthen skills in a changing, competitive, and highly demanding environment.

At TECH, you will experience a learning methodology that is shaking the foundations of traditional universities around the world"

You will have access to a learning system based on repetition, with natural and progressive teaching throughout the entire syllabus.

The student will learn to solve complex situations in real business environments through collaborative activities and real cases.

A learning method that is different and innovative

This TECH program is an intensive educational program, created from scratch, which presents the most demanding challenges and decisions in this field, both nationally and internationally. This methodology promotes personal and professional growth, representing a significant step towards success. The case method, a technique that lays the foundation for this content, ensures that the most current economic, social and professional reality is taken into account.

> *Our program prepares you to face new challenges in uncertain environments and achieve success in your career"*

The case method has been the most widely used learning system among the world's leading Information Technology schools for as long as they have existed. The case method was developed in 1912 so that law students would not only learn the law based on theoretical content. It consisted of presenting students with real-life, complex situations for them to make informed decisions and value judgments on how to resolve them. In 1924, Harvard adopted it as a standard teaching method.

What should a professional do in a given situation? This is the question that you are presented with in the case method, an action-oriented learning method. Throughout the course, students will be presented with multiple real cases. They will have to combine all their knowledge and research, and argue and defend their ideas and decisions.

Relearning Methodology

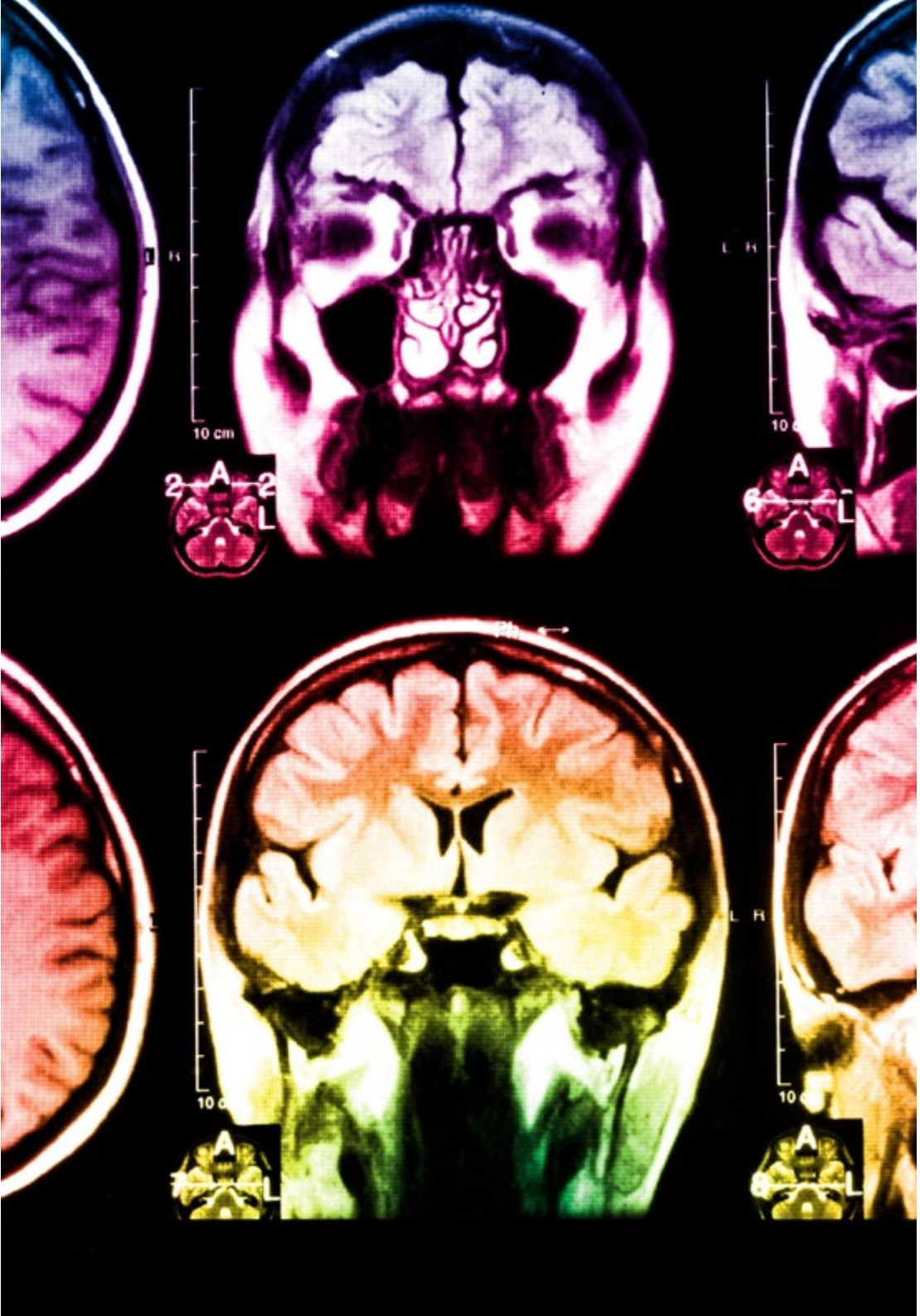
TECH effectively combines the Case Study methodology with a 100% online learning system based on repetition, which combines different teaching elements in each lesson.

We enhance the Case Study with the best 100% online teaching method: Relearning.

In 2019, we obtained the best learning results of all online universities in the world.

At TECH, you will learn using a cutting-edge methodology designed to train the executives of the future. This method, at the forefront of international teaching, is called Relearning.

Our university is the only one in the world authorized to employ this successful method. In 2019, we managed to improve our students' overall satisfaction levels (teaching quality, quality of materials, course structure, objectives...) based on the best online university indicators.



In our program, learning is not a linear process, but rather a spiral (learn, unlearn, forget, and re-learn). Therefore, we combine each of these elements concentrically. This methodology has trained more than 650,000 university graduates with unprecedented success in fields as diverse as biochemistry, genetics, surgery, international law, management skills, sports science, philosophy, law, engineering, journalism, history, and financial markets and instruments. All this in a highly demanding environment, where the students have a strong socio-economic profile and an average age of 43.5 years.

Relearning will allow you to learn with less effort and better performance, involving you more in your training, developing a critical mindset, defending arguments, and contrasting opinions: a direct equation for success.

From the latest scientific evidence in the field of neuroscience, not only do we know how to organize information, ideas, images and memories, but we know that the place and context where we have learned something is fundamental for us to be able to remember it and store it in the hippocampus, to retain it in our long-term memory.

In this way, and in what is called neurocognitive context-dependent e-learning, the different elements in our program are connected to the context where the individual carries out their professional activity.

This program offers the best educational material, prepared with professionals in mind:

Study Material

All teaching material is produced by the specialists who teach the course, specifically for the course, so that the teaching content is highly specific and precise.

These contents are then adapted in audiovisual format, to create the TECH online working method. All this, with the latest techniques that offer high-quality pieces in each and every one of the materials that are made available to the student.

Classes

There is scientific evidence suggesting that observing third-party experts can be useful.

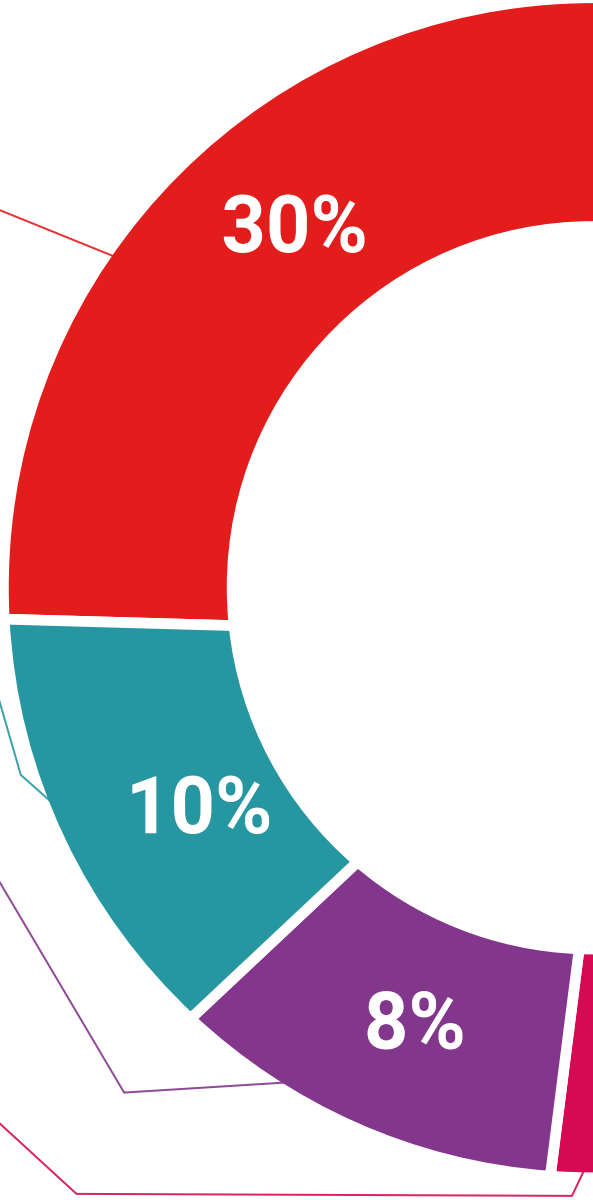
Learning from an Expert strengthens knowledge and memory, and generates confidence in future difficult decisions.

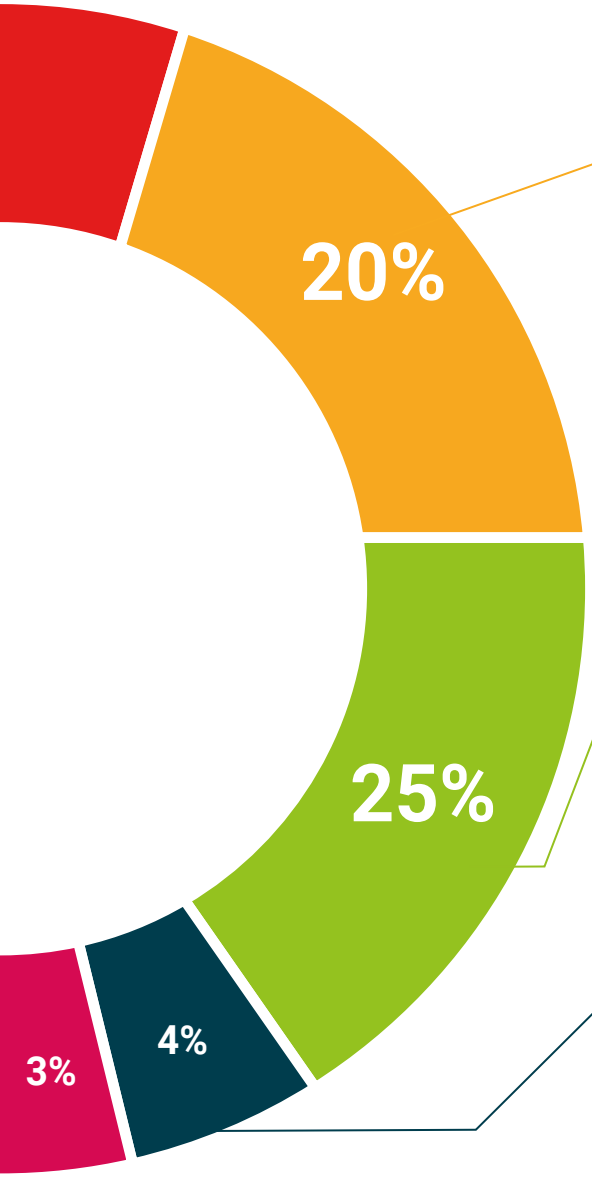
Practising Skills and Abilities

They will carry out activities to develop specific competencies and skills in each thematic area. Exercises and activities to acquire and develop the skills and abilities that a specialist needs to develop in the context of the globalization that we are experiencing.

Additional Reading

Recent articles, consensus documents and international guidelines, among others. In TECH's virtual library, students will have access to everything they need to complete their course.





Case Studies

Students will complete a selection of the best case studies chosen specifically for this program. Cases that are presented, analyzed, and supervised by the best specialists in the world.

Interactive Summaries

The TECH team presents the contents attractively and dynamically in multimedia lessons that include audio, videos, images, diagrams, and concept maps in order to reinforce knowledge.

This exclusive educational system for presenting multimedia content was awarded by Microsoft as a "European Success Story".

Testing & Retesting

We periodically evaluate and re-evaluate students' knowledge throughout the program, through assessment and self-assessment activities and exercises, so that they can see how they are achieving their goals.

06

Certificate

The Professional Master's Degree in Software Development guarantees students, in addition to the most rigorous and up-to-date education, access to a Professional Master's Degree issued by TECH Technological University.

Successfully complete this program and receive your university qualification without having to travel or fill out laborious paperwork"

This **Professional Master's Degree in Software Development** contains the most complete and up-to-date program on the market.

After the student has passed the assessments, they will receive their corresponding **Professional Master's Degree** issued by **TECH Technological University** via tracked delivery*.

The certificate issued by **TECH Technological University** will reflect the qualification obtained in the Professional Master's Degree, and meets the requirements commonly demanded by labor exchanges, competitive examinations, and professional career evaluation committees.

Title: **Professional Master's Degree in Software Development**

Official Nº of Hours: **1,500 h.**


tech technological university

Awards the following

CERTIFICATE

to

Mr./Ms. _____, with identification number _____.

For having successfully passed and accredited the following program

PROFESSIONAL MASTER'S DEGREE

in

Software Development

This is a qualification awarded by this University, equivalent to 1,500 hours, with a start date of dd/mm/yyyy and an end date of dd/mm/yyyy.

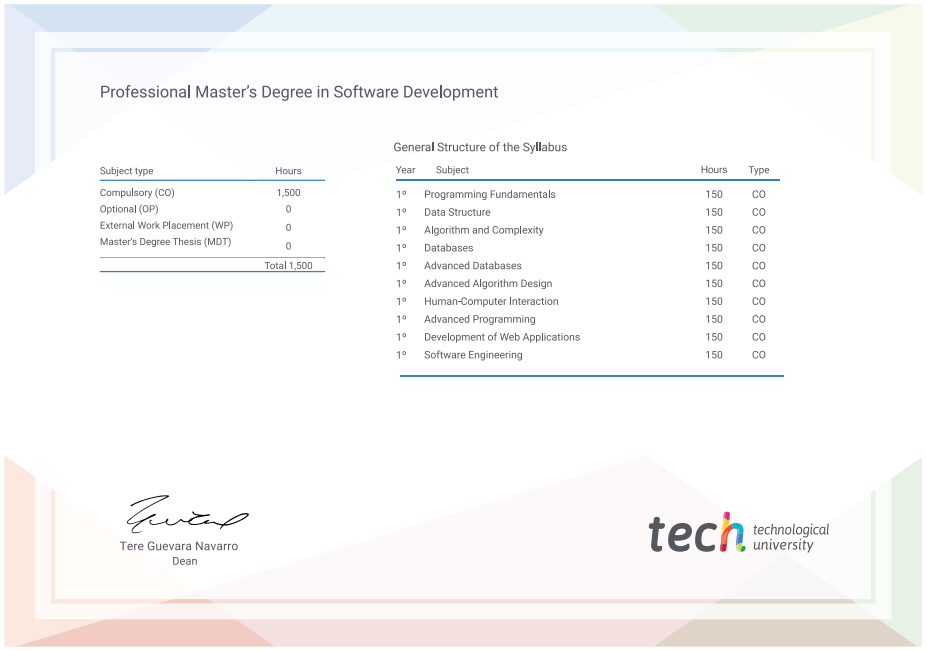
TECH is a Private Institution of Higher Education recognized by the Ministry of Public Education as of June 28, 2018.

June 17, 2020

Tere Guevara Navarro
Dean

This qualification must always be accompanied by the university degree issued by the competent authority to practice professionally in each country.

Unique TECH Code: AFWORD23S techtitute.com/certificates


Professional Master's Degree in Software Development

Subject type	Hours
Compulsory (CO)	1,500
Optional (OP)	0
External Work Placement (WP)	0
Master's Degree Thesis (MDT)	0
	Total 1,500

General Structure of the Syllabus

Year	Subject	Hours	Type
1º	Programming Fundamentals	150	CO
1º	Data Structure	150	CO
1º	Algorithm and Complexity	150	CO
1º	Databases	150	CO
1º	Advanced Databases	150	CO
1º	Advanced Algorithm Design	150	CO
1º	Human-Computer Interaction	150	CO
1º	Advanced Programming	150	CO
1º	Development of Web Applications	150	CO
1º	Software Engineering	150	CO

Tere Guevara Navarro
Dean

tech technological university

*Apostille Convention. In the event that the student wishes to have their paper certificate issued, with an apostille, TECH EDUCATION will make the necessary arrangements to obtain it, at an additional cost.

tech technological university

Professional Master's Degree
Software Development

- Modality: **online**
- Duration: **12 months**
- Certificate: **TECH Technological University**
- Dedication: **16h/week**
- Schedule: **at your own pace**
- Exams: **online**

Professional Master's Degree
Software Development

tech technological university