Hybrid Professional Master's Degree
Software Development

tech global university

Hybrid Professional Master's Degree
Software Development

Modality: **Hybrid (Online + Internship)**
Duration: **12 months**
Certificate: **TECH Global University**
Accreditation: **60 + 4 ECTS credits**
Website: www.techtitute.com/us/information-technology/hybrid-professional-master-degree/hybrid-professional-master-degree-software-development

Index

Introduction

Software Development has become a crucial component of the technology infrastructure in most industries, from healthcare to finance to entertainment. With the increasing complexity of applications, IT must adapt to a constantly changing environment. This requires developers to update their knowledge frequently to keep abreast of the latest developments in subjects such as programming languages or strategies for creating algorithms. In this context, TECH presents an innovative university program that will delve into the most recent innovations in the field of Software Development.

Thanks to this Hybrid Professional Master's Degree, you will apply design patterns to solve a wide range of IT problems and build scalable solutions"

Software Development has become an essential factor in today's digital economy, with an increasing demand for skilled professionals worldwide. According to the Organization for Economic Cooperation and Development, this professional sector is expected to experience 30% growth over the next year. This underscores the importance of IT professionals staying abreast of the latest trends in the field. Otherwise, developers could face problems such as the use of outdated methodologies that lead to less efficient processes.

Within this framework, TECH launches a revolutionary Hybrid Professional Master's Degree in Software Development. It is a university program that combines in 1,920 hours the best theoretical content with 3 weeks of practical stay in a reference entity in this field. The academic itinerary will delve into aspects such as C++ programming, the creation of advanced databases or the architectural design of applications. All of this is done through teaching materials prepared by an experienced teaching staff, which include a wide range of multimedia resources (such as interactive summaries, case studies or explanatory videos) to ensure an enjoyable refresher course. In this way, computer scientists will enjoy a totally progressive and natural learning process, without having to resort to traditional techniques such as memorization.

In addition, the university program provides for graduates to undertake practical training at a prestigious institution. There, the computer scientists will actively participate in the projects that are being developed at the time. It should be noted that a specialized tutor will guide the students during this on-site stay, ensuring the completion of a plan of activities that will allow them to optimize their skills based on the demands of today's labor market.

This **Hybrid 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 more than 100 practical cases presented by Software Development professionals
- Its graphic, schematic and practical contents, with which they are conceived, gather essential information on Software Development
- Updates on the latest developments in Software Development
- It contains practical exercises where the self-evaluation process can be carried out to improve learning
- 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
- Furthermore, you will be able to carry out an internship in one of the best companies

You will implement in your practice the quality assurance processes to guarantee the reliability and performance of the Software”

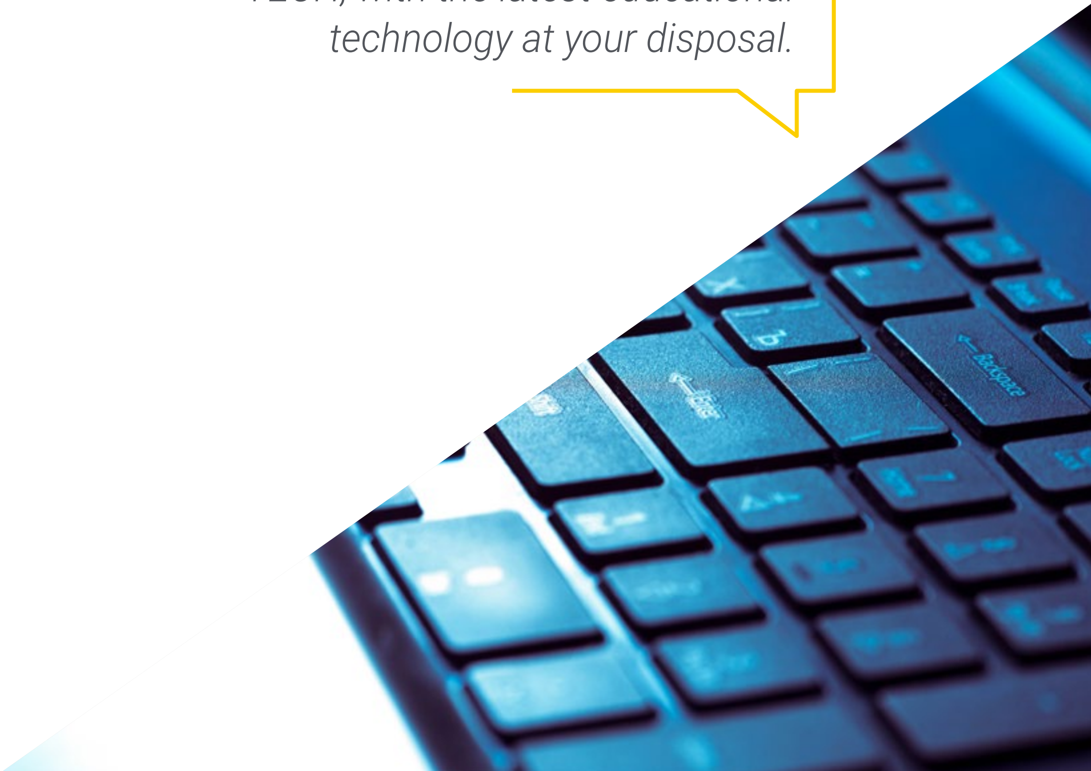
Take an intensive 3-week internship in a prestigious company and acquire all the knowledge to experience a considerable leap in professional quality"

This university degree will include real cases to bring the development of the program as close as possible to the reality of IT practice.

You will have the support of the largest online academic institution in the world, TECH, with the latest educational technology at your disposal.

In this Hybrid Professional Master's Degree proposal, of a professionalizing nature and blended learning modality, the program is aimed at updating IT professionals who want to incorporate the most advanced techniques for Software Development into their practice. The contents are based on the latest scientific evidence, and oriented in a didactic way to integrate theoretical knowledge into practice, and the theoretical-practical elements will facilitate the updating of knowledge.

Thanks to its multimedia content elaborated with the latest educational technology, it will allow the IT professional a situated and contextual learning, that is to say, a simulated environment that will provide an immersive learning programmed to specialize in real situations. This program is designed around Problem-Based Learning, whereby the physician must try to solve the different professional practice situations that arise during the course. For this purpose, the students will be assisted by an innovative interactive video system created by renowned and experienced experts.



02

Why Study this Hybrid Professional Master's Degree?

The demand for Software Developers is constantly growing, due to technological progress. In fact, experts predict that this professional profile will grow by 30% over the next few years. In view of this, computer scientists need to incorporate into their practice the most innovative methodologies for the development of networked applications. For this reason, TECH has created this pioneering degree, which combines the most recent updates in areas such as Software Engineering or algorithm creation with a practical stay in a reference entity. In this way, graduates will obtain advanced skills to offer high quality services.

A high intensity curriculum that will lay the foundations for your professional growth and place you at the pinnacle of Computer Science"

1. Updating from the latest technology available

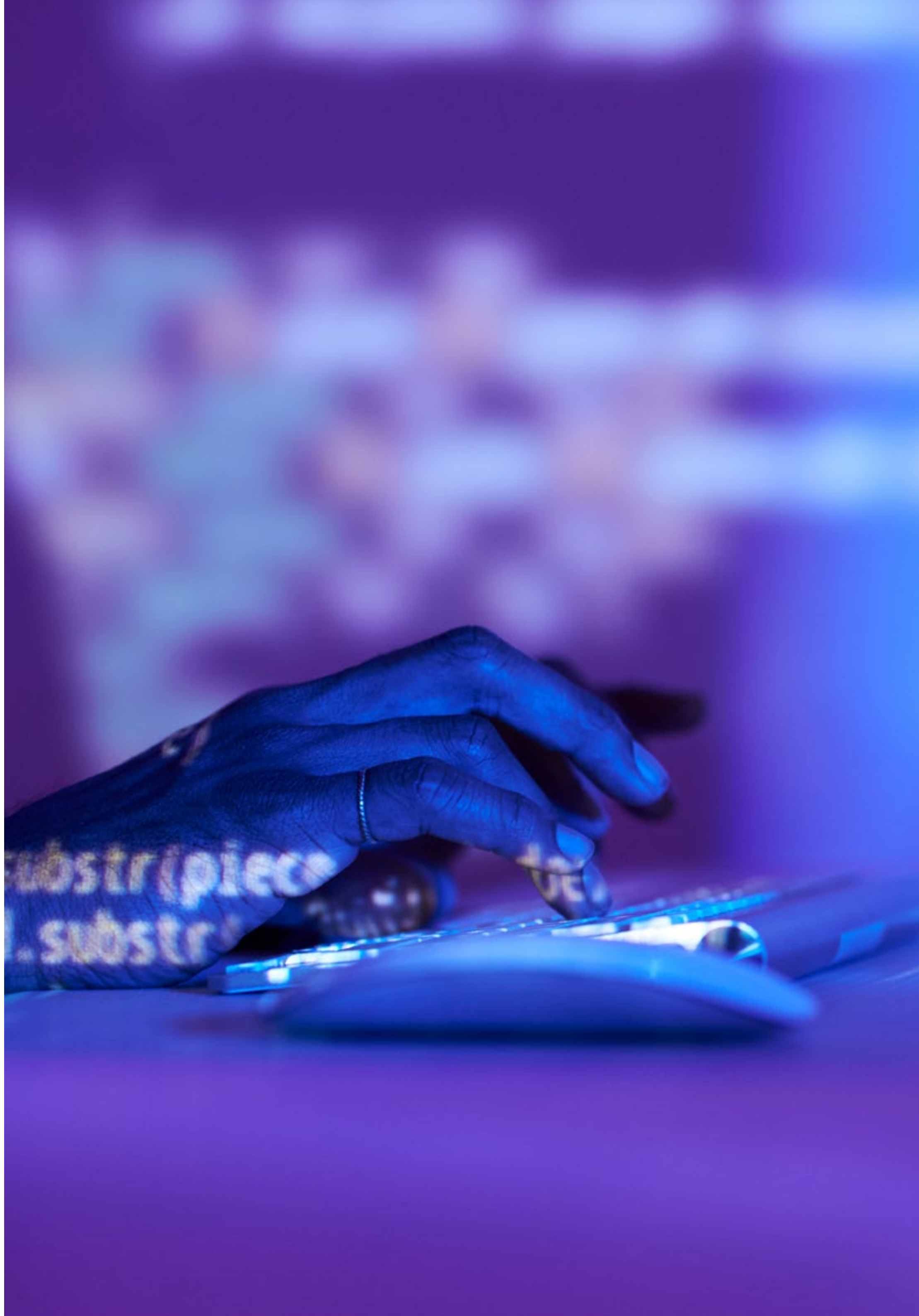
New technologies are significantly transforming software development, improving productivity, efficiency and quality. These tools allow computer scientists to address more complex challenges, create more robust applications and adapt quickly to the changing needs of the market. In this context, TECH presents this Hybrid Professional Master's Degree, which will provide students with the most sophisticated tools to perform their tasks effectively.

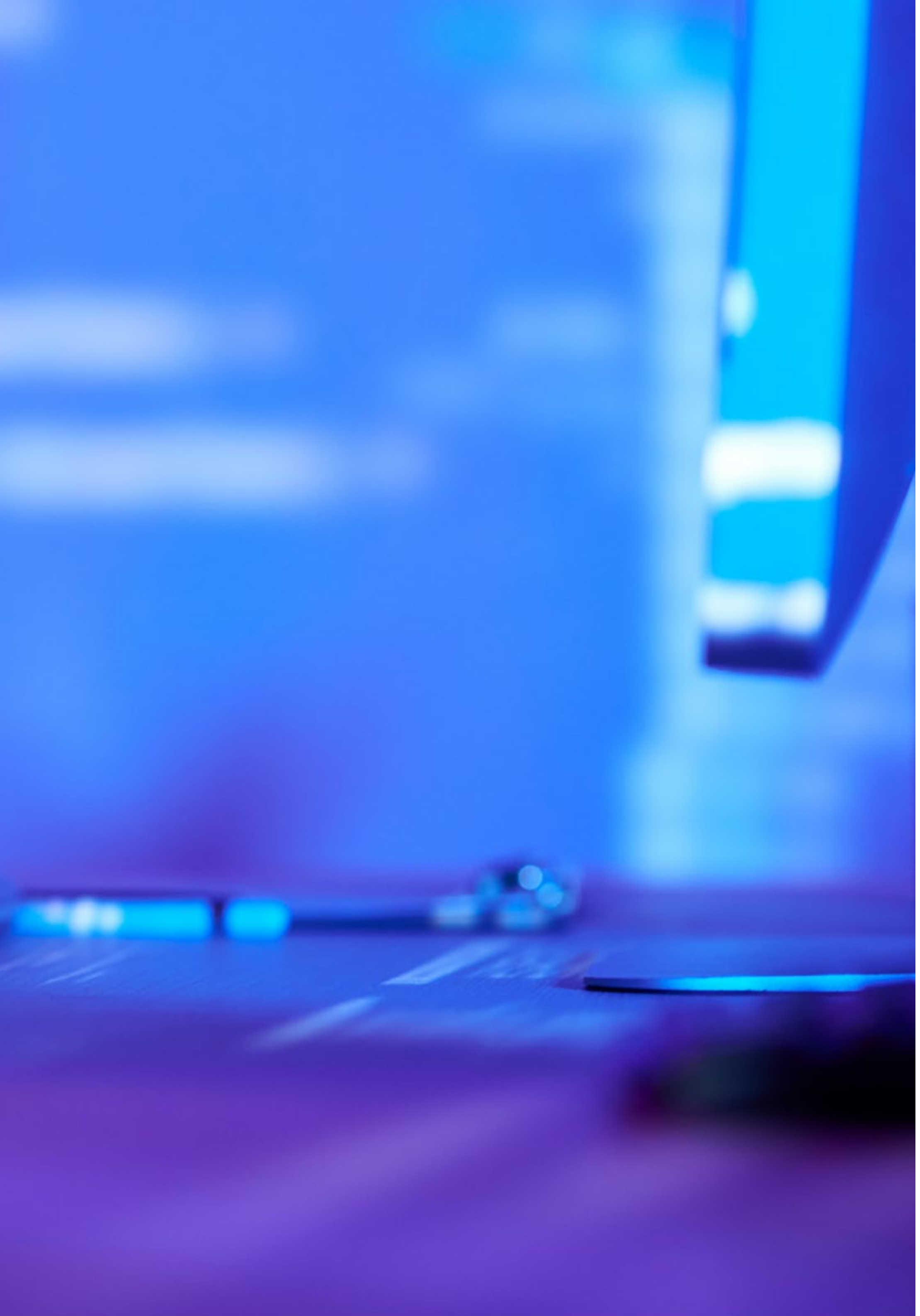
2. Gaining in-depth knowledge from the experience of top specialists

Throughout the practical period, a team of Software Development professionals will accompany students to help them get the most out of this academic experience. At the same time, they will pass on the most innovative techniques to create the most complete and accessible software architectures.

3. Entering first-class professional environments

TECH's main premise is to make first class university programs available to everyone. For this reason, TECH carefully selects all the centers available for internships. Thanks to this effort, computer scientists will have access to reference institutions in the field of Software Development. In this way, you will be able to experience the day-to-day work of a demanding, rigorous and exhaustive work area, always applying the latest techniques in its work methodology.





4. Combining the best theory with state-of-the-art practice

The academic market is full of pedagogical degrees that are limited to providing theoretical content, forgetting that practice is a fundamental aspect for students to apply their knowledge to real work situations. Far from this, TECH offers a 100% practical learning model, which will allow graduates to acquire practical experience and face the real challenges they may encounter in their professional career.

5. Expanding the boundaries of knowledge

TECH offers graduates the opportunity to carry out this Hybrid Professional Master's Degree Program in international organizations. Thanks to this, computer scientists will be able to expand their frontiers and catch up with the best professionals who work in top-level companies. A unique opportunity that only TECH, the world's largest online university could offer.

You will have full practical immersion at the center of your choice"

03

Objectives

Thanks to this university degree, computer science professionals will master modern programming languages such as C++. Graduates will also gain the skills to write clean, efficient and maintainable code. In this way, developers will design software architectures that support the scalability, flexibility and integrity of systems.

This university program will provide you with the most cutting-edge strategies to create highly efficient Databases"

General Objective

- This Hybrid Professional Master's Degree in Software Development will equip computer scientists with both the knowledge and practical skills necessary to meet the challenges in this field. Graduates will effectively apply design patterns to solve common problems and build robust software solutions. They will also be able to mitigate vulnerabilities in programs by implementing secure development practices. In addition, students will create and manage relational databases to meet information storage and retrieval needs

Reach your full potential in the field of Software Development thanks to the most complete teaching materials in the academic market"

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, and 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 basics of programming in the C++ language, 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 the operation of hash tables, such 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 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 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
- Laying the groundwork for understanding the operation of NoSQL databases, as well as introducing the Mongo DB 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 Algorithms Design

- Delve into advanced algorithm design, analyzing recursive and divide-and-conquer algorithms, as well as performing amortized analysis
- Understand dynamic programming concepts 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 evaluate them
- Understand the different methods of interaction with the computer, by means of peripherals and devices

Module 8. Advanced Programming

- Delve into the 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 user interface development 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 how to use the browser-oriented programming language JavaScript, and some of its main features
- Understand the concepts of component-oriented programming and the component architecture
- Learn how to use the Bootstrap front-end 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

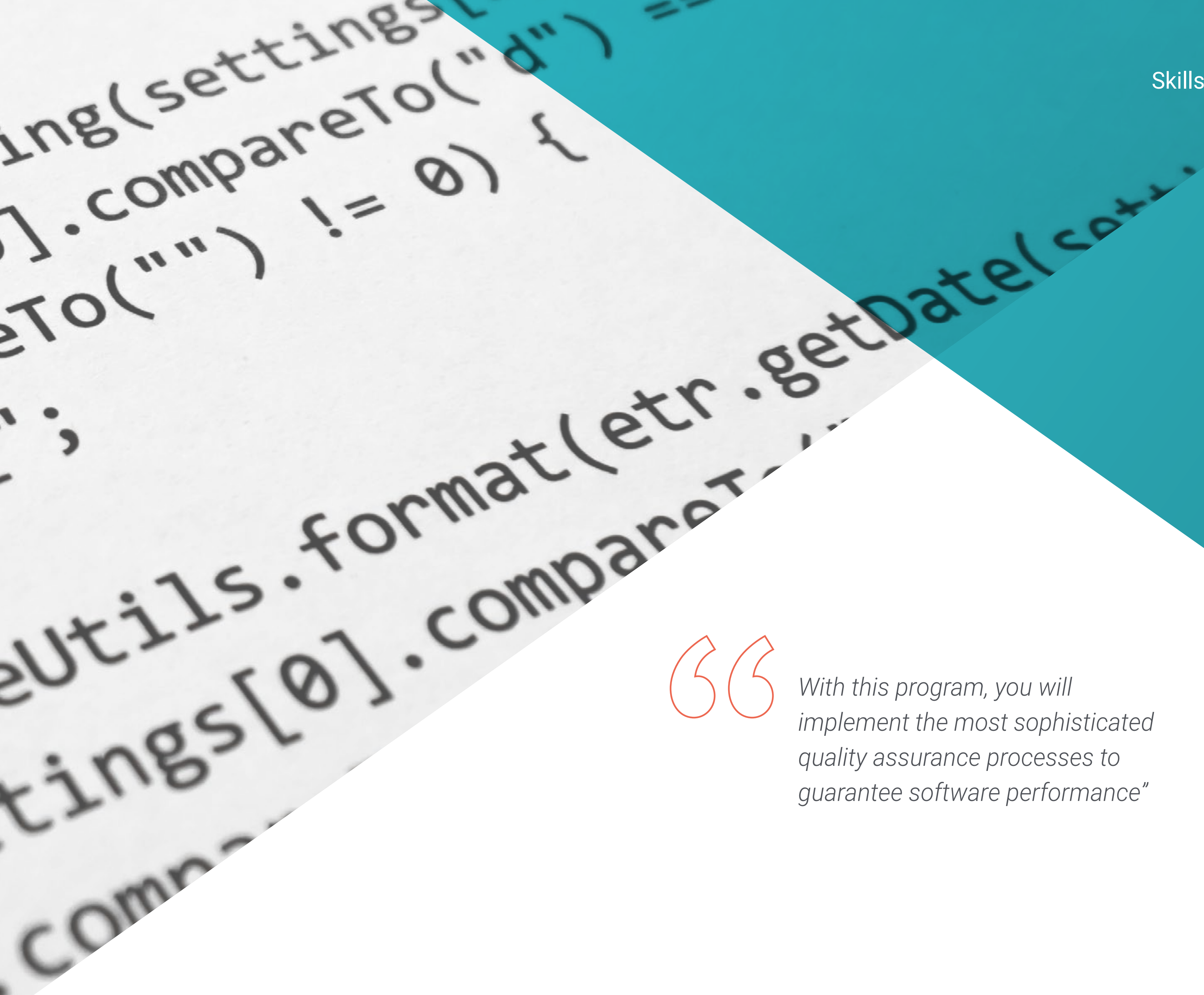
- Lay the foundations of software engineering and modeling, learning the main processes and concepts
- Understand the software process and the different models for its development including agile technologies
- Understand requirements engineering, their development, elaboration, negotiation and validation
- 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

“ *You will combine theory and professional practice through a demanding and rewarding educational approach"*

Skills

Upon completion of this university degree, computer scientists will acquire advanced skills to overcome challenges in the field of Software Development. Similarly, graduates will master programming languages such as C++, which will enable them to create high-performance applications. Likewise, developers will implement agile methodologies (such as Scrum) to their projects to optimize the flexibility and responsiveness of software.

```
etr.getStr
se if (settings[0
if (name.compare
name += "
}
name += Date
if (set
```

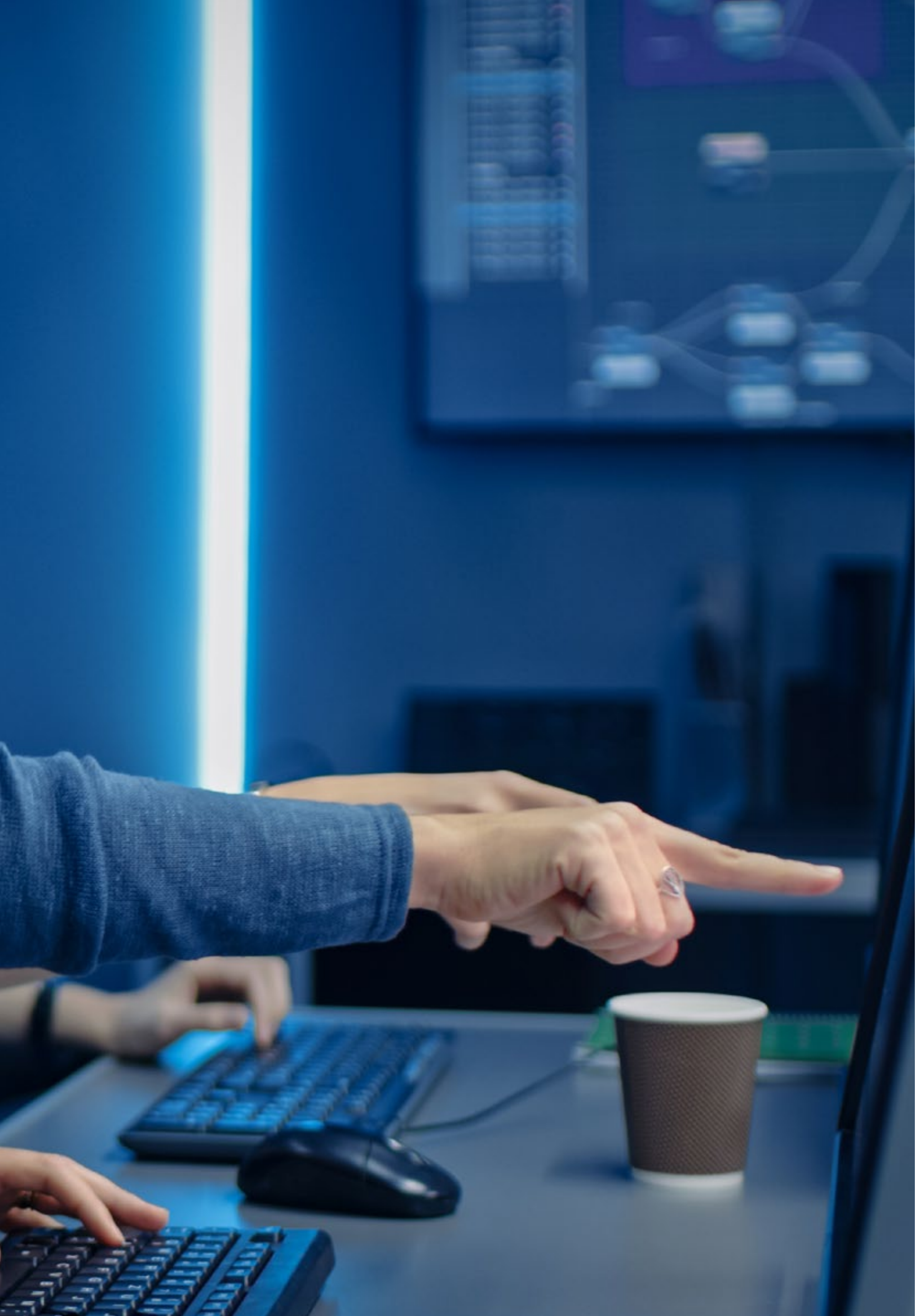


With this program, you will implement the most sophisticated quality assurance processes to guarantee software performance"

General Skills

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

This university program offered you the opportunity to extensive your knowledge in a real scenario, with the maximum scientific rigor of an institution at the forefront of technology"



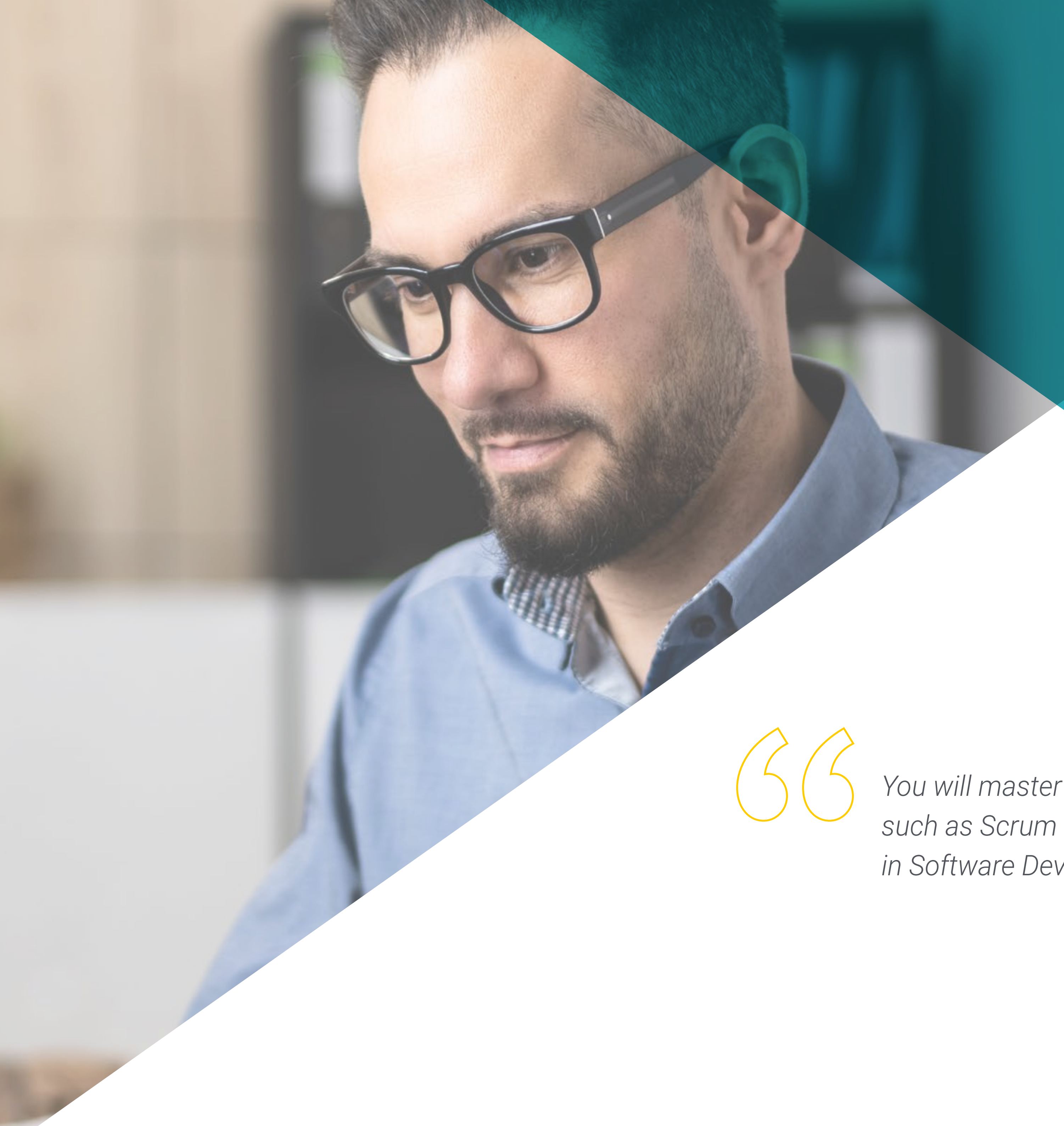
Specific Skills

- 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

05

Educational Plan

The teaching materials that make up this Hybrid Professional Master's Degree have been developed by true experts in the field of Software Development. Composed of 10 specialized modules, the program will equip computer scientists with a wide range of technical skills. The syllabus will delve into aspects such as Algorithm Design, Database creation or Software Engineering. In addition, the syllabus will provide developers with the most innovative techniques for resource optimization, among which Backtracking stands out. In this way, graduates will acquire advanced skills to design software architectures that support the growth and evolution of systems.



You will master agile methodologies such as Scrum to improve efficiency in Software Development"

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 Algorithms 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: 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. Web Application Development

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. 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. 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. Software Engineering Guiding Principles
- 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. Software Quality
- 10.10.2. The Software Quality Dilemma
- 10.10.3. Achieving Software Quality
- 10.10.4. Ensuring Software Quality
- 10.10.5. The Administrative Spectrum
- 10.10.6. The Staff
- 10.10.7. The product
- 10.10.8. The Process
- 10.10.9. The Project
- 10.10.10. Principles and Practices

06

Clinical Internship

Once the online theoretical period has been completed, this university program includes a practical training phase in a reference entity. During this period, graduates will have the support of a tutor who will help them to get the most out of this experience. Thanks to this, the computer scientist will acquire advanced skills to experience a remarkable leap in quality in the practice of their profession.

Through this university degree, you will be prepared to overcome the challenges in the field of Software Development"

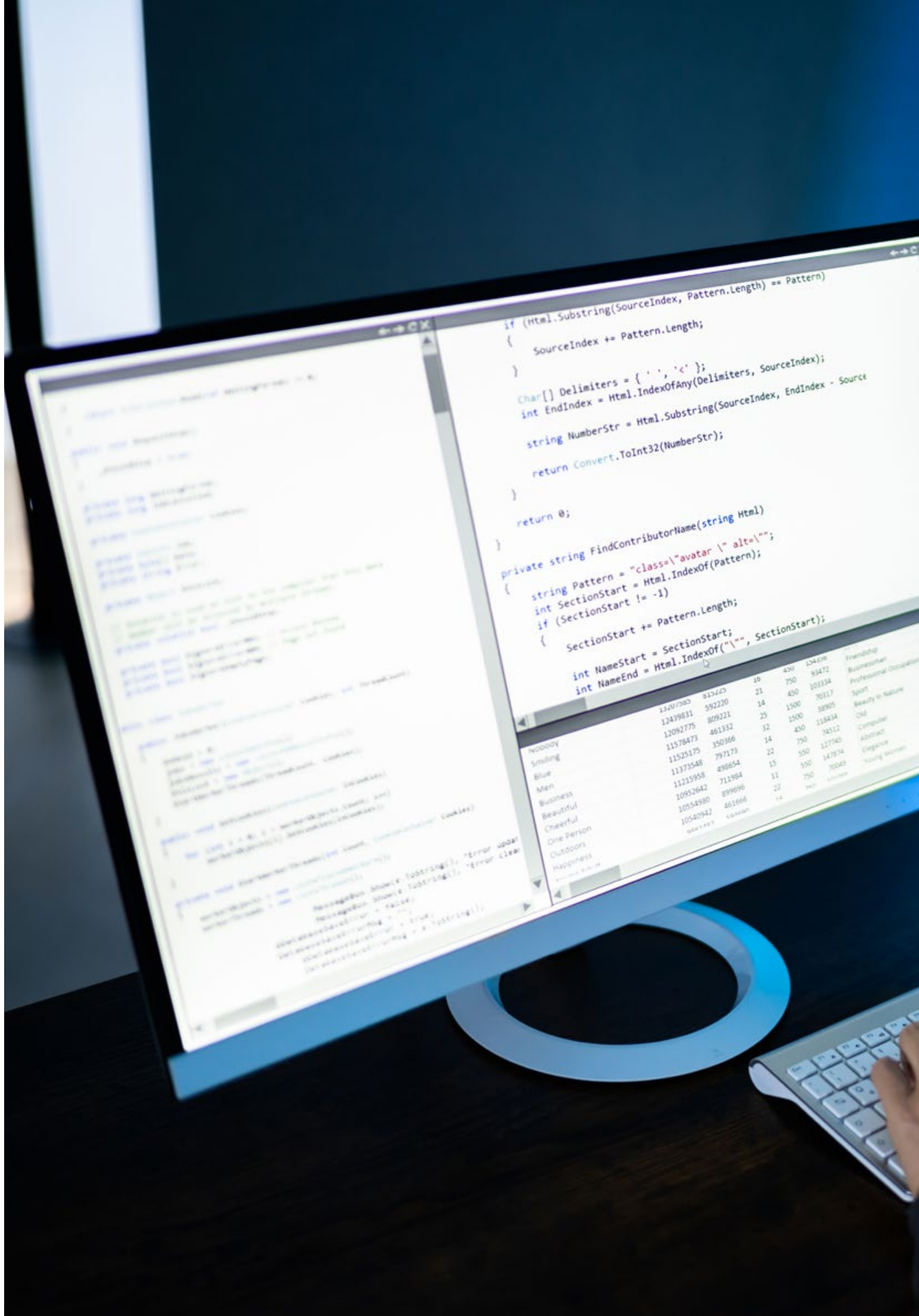
The Internship Program of this program in Software Development consists of an intensive practical internship in a prestigious company, lasting 3 weeks, from Monday to Friday, with 8 consecutive hours of practical training, with an assistant specialist. In this way, this stay will allow the professionals to apply the theoretical concepts learned in real work situations, either in companies of the sector or in collaborative projects.

In this program proposal, completely practical in nature, the activities are aimed at developing and perfecting the necessary skills to provide Software Development services, as well as conditions that require a high level of qualification, oriented to the specific training for the exercise of the activity.

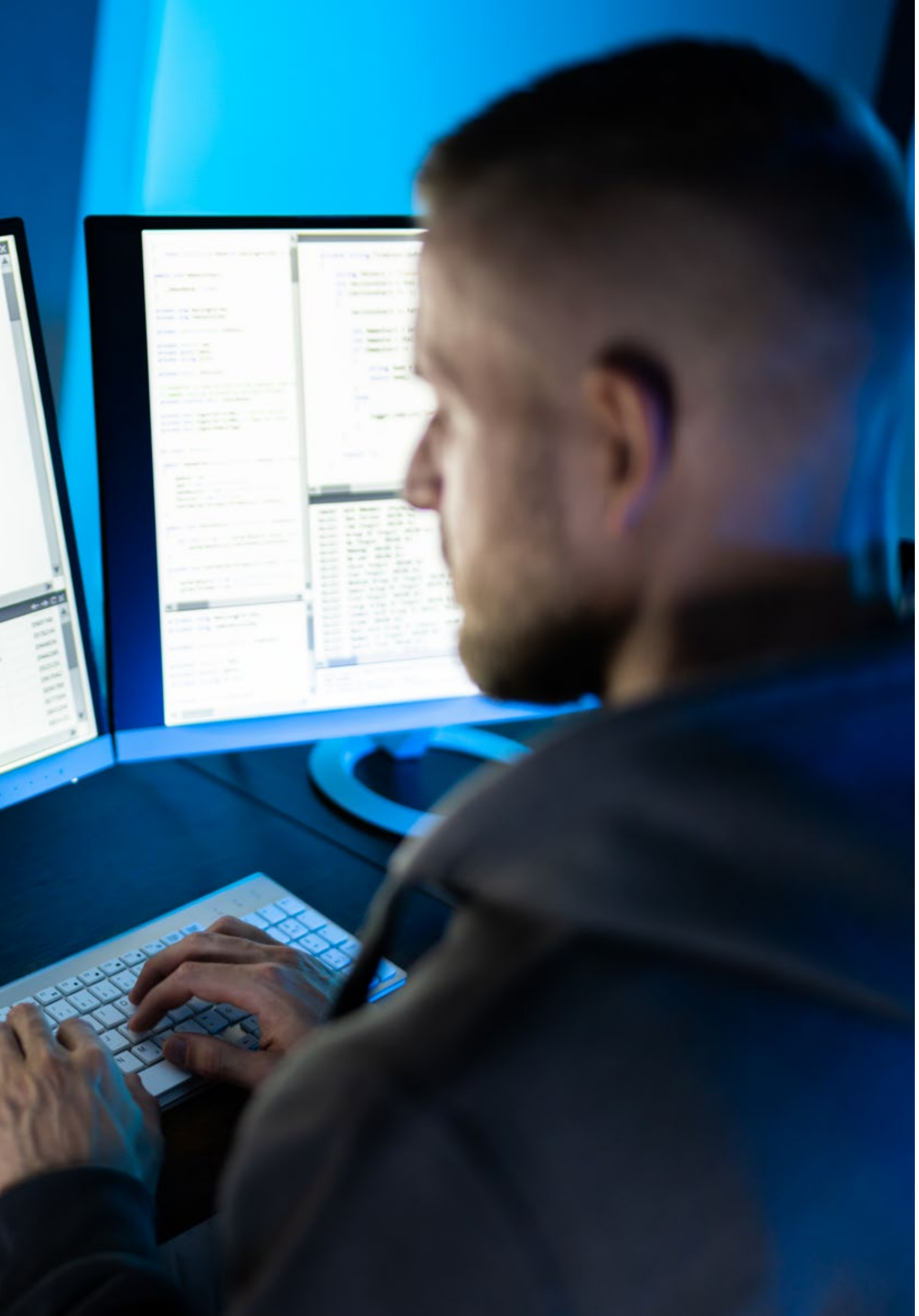
Undoubtedly, this is an excellent opportunity for graduates to immerse themselves in the reality of a profession full of challenges. In this way, they will collaborate in projects related to software design, database creation or algorithm development, among others. Therefore, the computer scientists will develop advanced skills to optimize their practice and raise their professional horizons.

The practical education will be carried out with the active participation of the student performing the activities and procedures of each area of competence (learning to learn and learning to do), with the accompaniment and guidance of teachers and other training partners that facilitate teamwork and multidisciplinary integration as transversal competencies for the praxis of Software Development (learning to be and learning to relate).

The procedures described below will be the basis of the practical part of the



```
if (Html.Substring(SourceIndex, Pattern.Length) == Pattern)
{
    SourceIndex += Pattern.Length;
}
Char[] Delimiters = { ' ', '<' };
int EndIndex = Html.IndexOfAny(Delimiters, SourceIndex);
string NumberStr = Html.Substring(SourceIndex, EndIndex - Source
return Convert.ToInt32(NumberStr);
}
return 0;
}
private string FindContributorName(string Html)
{
    string Pattern = "class=\"avatar \" alt=\"";
    int SectionStart = Html.IndexOf(Pattern);
    if (SectionStart != -1)
    {
        SectionStart += Pattern.Length;
        int NameStart = SectionStart;
        int NameEnd = Html.IndexOf("\"", SectionStart);
```



program, and their implementation will be subject to the center's own availability and workload, the proposed activities being the following:

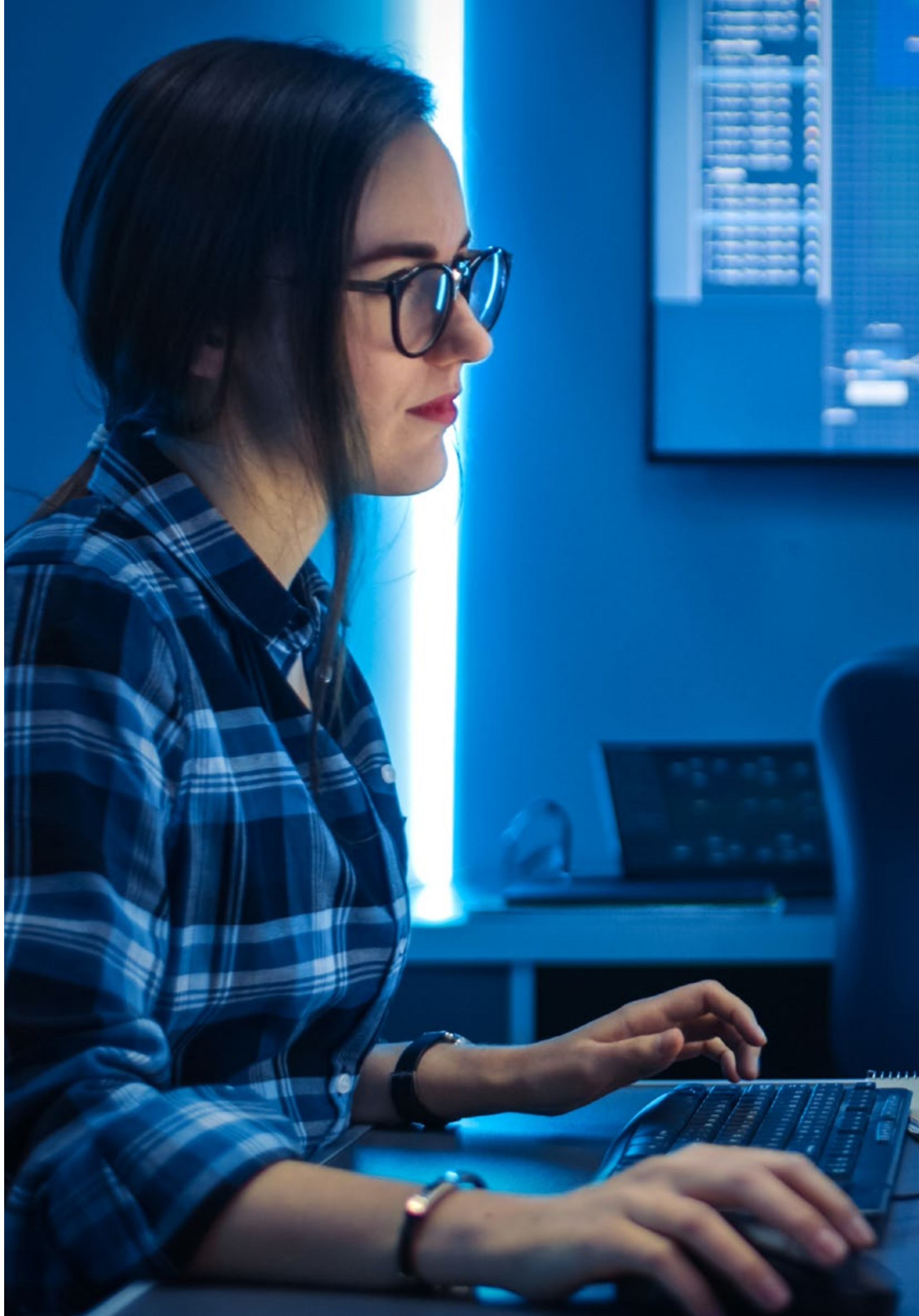
Module	Practical Activity
Data Architecture	Create new data structures that are efficient and suitable for solving specific problems.
	Implement basic data structures such as pulses, queues, trees, graphs, etc.
	Evaluate the time complexity of different algorithm structures
	Realize efficient database schemas employing data systems that optimize storage and retrieval
Algorithm Techniques	Design algorithms in different programming languages.
	Use advanced techniques such as Dynamic Programming and Graph Algorithms
	Develop algorithms that minimize the use of computational resources such as memory and CPU time
	Test algorithms to verify their correct operation in different scenarios and with different datasets
Data Storage Systems	Configure databases in management systems such as MySQL, PostgreSQL, etc.
	Use monitoring tools to monitor database performance to ensure database reliability and availability
	Implement access controls and security policies in order to protect data from unauthorized access
	Execute database migration from one environment to another (e.g. from a local database to one in the cloud)
Software Development	Carry out system architecture, including division into modules and specification of interfaces
	Develop detailed designs of each component or module of the system, including UML diagrams and technical specifications.
	Perform unit tests to verify the correct functioning of individual code modules.
	Identify and correct bugs in the software after deployment, to implement new functionalities or enhancements.-

Civil Liability Insurance

This institution's main concern is to guarantee the safety of the students and other collaborating agents involved in the internship process at the company. Among the measures dedicated to achieve this is the response to any incident that may occur during the entire teaching-learning process.

To this end, this entity commits to purchasing a civil liability insurance policy to cover any eventuality that may arise during the course of the internship at the center.

This liability policy for interns will have broad coverage and will be taken out prior to the start of the practical training period. That way professionals will not have to worry in case of having to face an unexpected situation and will be covered until the end of the internship program at the center.



General Conditions of the Internship Program

The general terms and conditions of the internship agreement for the program are as follows:

1. TUTOR: During the Hybrid Professional Master's Degree, students will be assigned with two tutors who will accompany them throughout the process, answering any doubts and questions that may arise. On the one hand, there will be a professional tutor belonging to the internship center who will have the purpose of guiding and supporting the student at all times. On the other hand, they will also be assigned with an academic tutor whose mission will be to coordinate and help the students during the whole process, solving doubts and facilitating everything they may need. In this way, the student will be accompanied and will be able to discuss any doubts that may arise, both clinical and academic.

2. DURATION: The internship program will have a duration of three continuous weeks, in 8-hour days, 5 days a week. The days of attendance and the schedule will be the responsibility of the center and the professional will be informed well in advance so that they can make the appropriate arrangements.

3. ABSENCE: If the students does not show up on the start date of the Hybrid Professional Master's Degree, they will lose the right to it, without the possibility of reimbursement or change of dates. Absence for more than two days from the internship, without justification or a medical reason, will result in the professional's withdrawal from the internship, therefore, automatic termination of the internship. Any problems that may arise during the course of the internship must be urgently reported to the academic tutor.

4. CERTIFICATION: Professionals who pass the Hybrid Professional Master's Degree will receive a certificate accrediting their stay at the center.

5. EMPLOYMENT RELATIONSHIP: the Hybrid Professional Master's Degree shall not constitute an employment relationship of any kind.

6. PRIOR EDUCATION: Some centers may require a certificate of prior education for the Hybrid Professional Master's Degree. In these cases, it will be necessary to submit it to the TECH internship department so that the assignment of the chosen center can be confirmed.

7. DOES NOT INCLUDE: The Hybrid Professional Master's Degree will not include any element not described in the present conditions. Therefore, it does not include accommodation, transportation to the city where the internship takes place, visas or any other items not listed.

However, students may consult with their academic tutor for any questions or recommendations in this regard. The academic tutor will provide the student with all the necessary information to facilitate the procedures in any case.

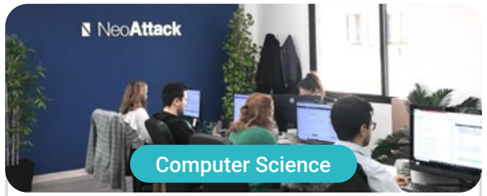
07

Where Can I Do the Internship?

In line with its commitment to provide a high quality education accessible to all, TECH expands the academic opportunities for students and enables the practical stay to be carried out in various prestigious international entities. Therefore, graduates have an ideal opportunity to improve their professional quality by working with the best specialists in the field of Software Development.

You will complete your learning in Software Development with a practical experience with professionals in the sector"

The student will be able to complete the practical part of this Hybrid Professional Master's Degree at the following centers:



NeoAttack

Country	City
Spain	Madrid

Address: Calle Santa Engracia 151, Planta 1, 1, Madrid

NeoAttack leads the market in carrying out SEO and advertising strategies.

Related internship programs:
Graphic Design
- Software Development

Captia Ingeniería

Country	City
Spain	Madrid

Address: Av. de las Nieves, 37, Bloque A Planta 1 Oficina E, 28935, Móstoles, Madrid

IT company dedicated to providing advanced technological solutions to industries.

Related internship programs:
- Visual Analytics and Big Data
- Software Development

"Boost your career path with holistic teaching, allowing you to advance both theoretically and practically"

08

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

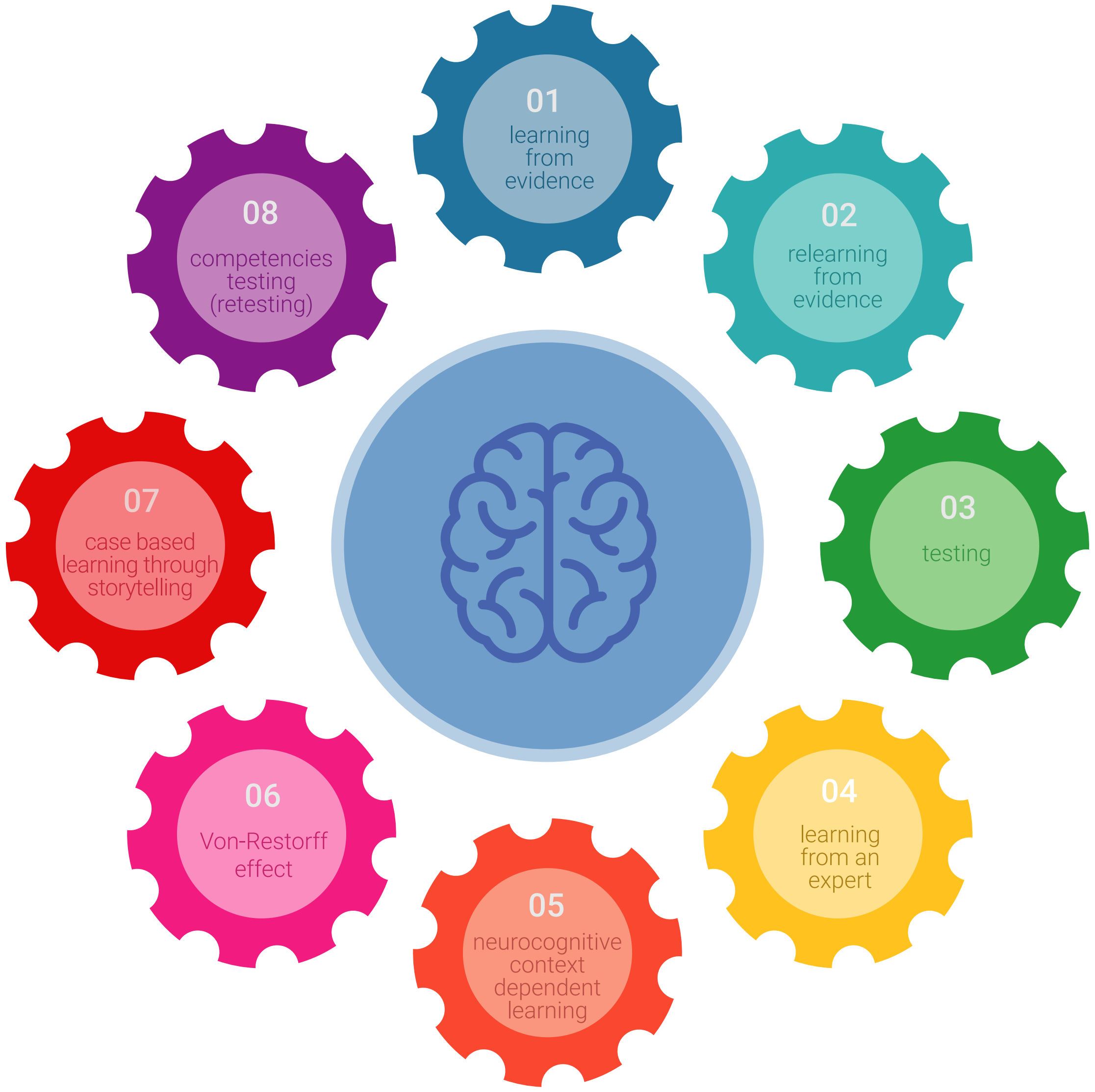
TECH effectively combines the Case Study methodology with a 100% online learning system based on repetition, which combines different teaching elements in each lesson.

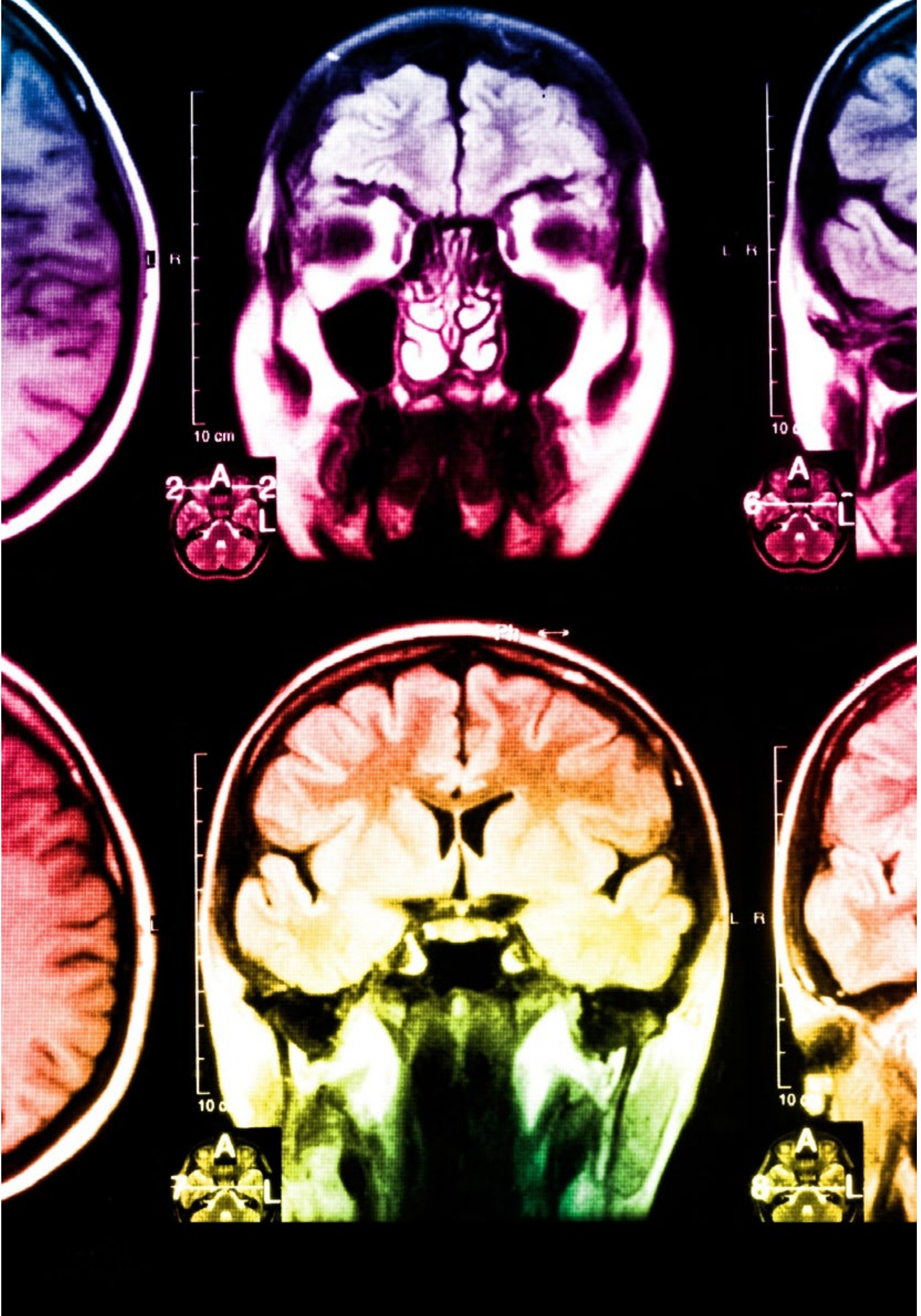
We enhance the Case Study with the best 100% online teaching method: Relearning.

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.


01 learning from evidence
02 relearning from evidence
03 testing
04 learning from an expert
05 neurocognitive context dependent learning
06 Von-Restorff effect
07 case based learning through storytelling
08 competencies testing (retesting)



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 applied to the 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.

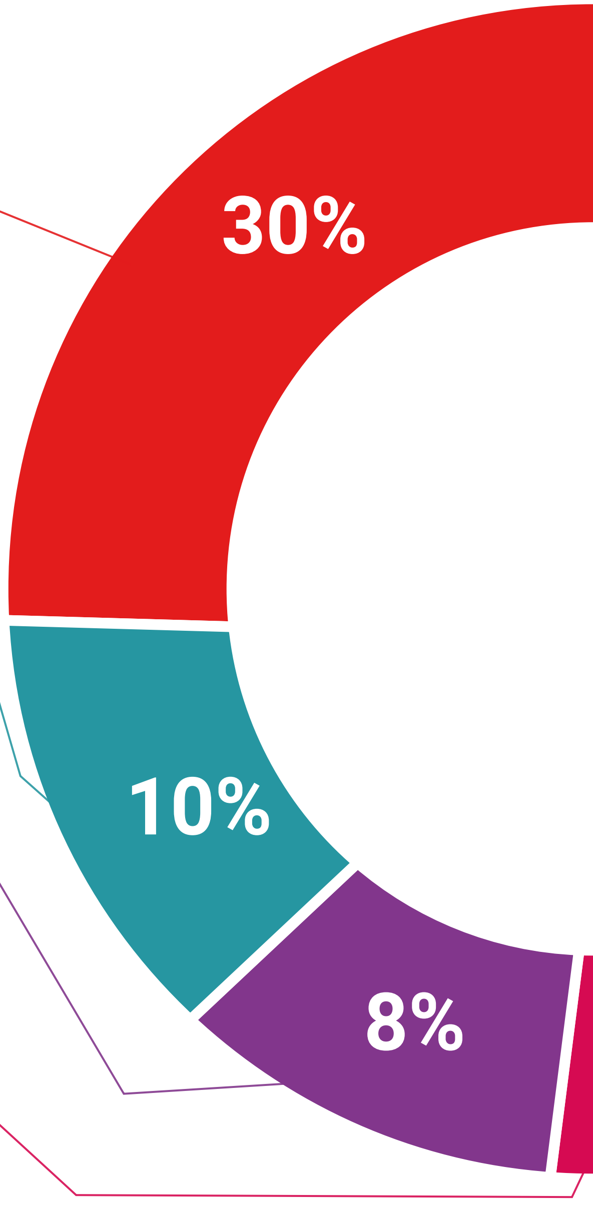
Learning from an Expert strengthens knowledge and memory, and generates confidence in future difficult decisions.

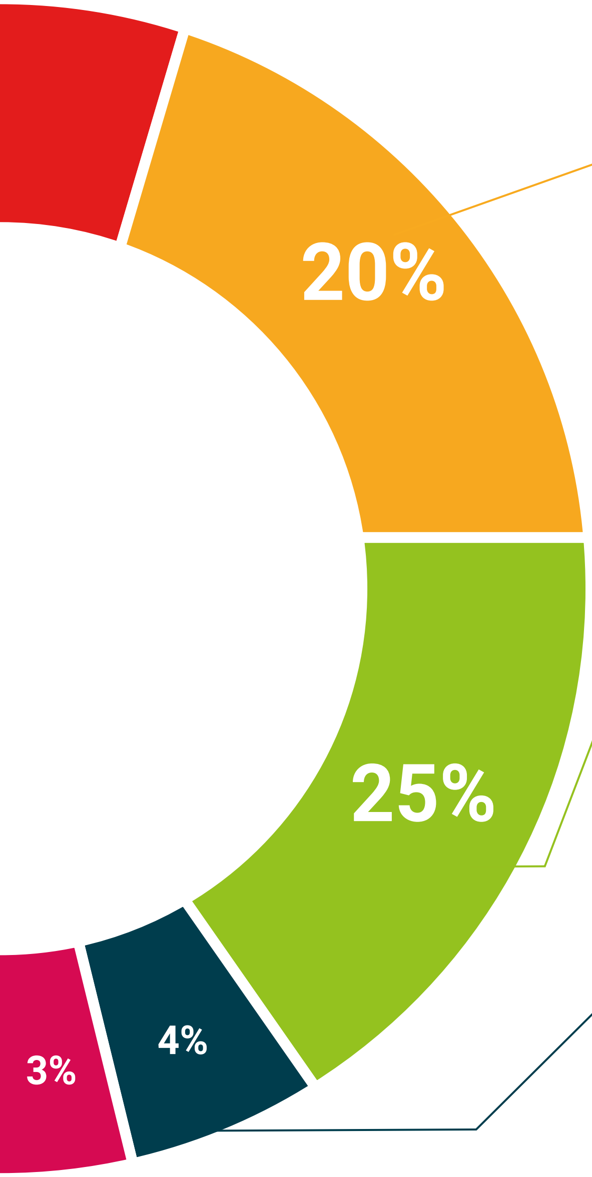
Practising Skills and Abilities

They will carry out activities to develop specific skills and abilities in each subject 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.

09

Certificate

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

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

This private qualification will allow you to obtain a **Hybrid Professional Master's Degree in Software Development** endorsed by **TECH Global University**, the world's largest online university.

TECH Global University is an official European University publicly recognized by the Government of Andorra (***official bulletin***). Andorra is part of the European Higher Education Area (EHEA) since 2003. The EHEA is an initiative promoted by the European Union that aims to organize the international training framework and harmonize the higher education systems of the member countries of this space. The project promotes common values, the implementation of collaborative tools and strengthening its quality assurance mechanisms to enhance collaboration and mobility among students, researchers and academics.

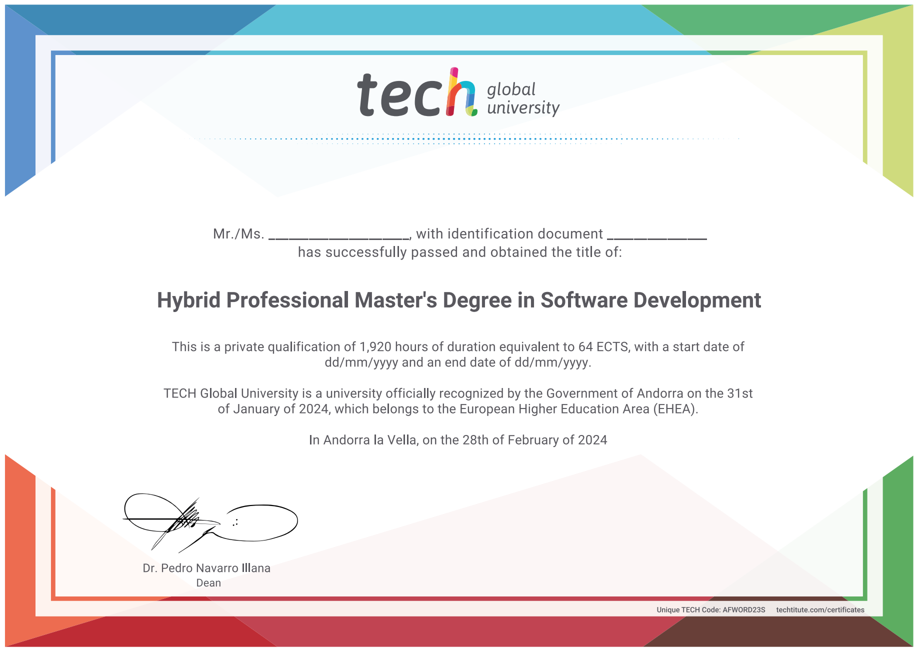
This **TECH Global University** private qualification is a European program of continuing education and professional updating that guarantees the acquisition of competencies in its area of knowledge, providing a high curricular value to the student who completes the program.

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

Modality: **Hybrid (Online + Internship)**

Duration: **12 months**

Accreditation: **60 + 4 ECTS**



Mr./Ms. ________________, with identification document ___________
has successfully passed and obtained the title of:

Hybrid Professional Master's Degree in Software Development

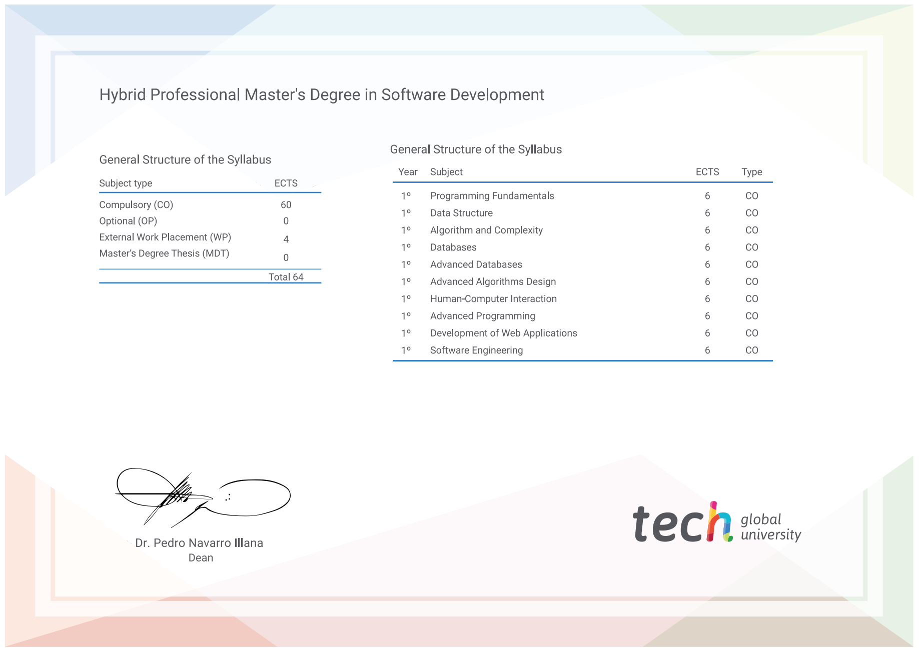
This is a private qualification of 1,920 hours of duration equivalent to 64 ECTS, with a start date of dd/mm/yyyy and an end date of dd/mm/yyyy.

TECH Global University is a university officially recognized by the Government of Andorra on the 31st of January of 2024, which belongs to the European Higher Education Area (EHEA).

In Andorra la Vella, on the 28th of February of 2024

[illegible]
Dr. Pedro Navarro Illana
Dean

Unique TECH Code: AFWORD23S techtitute.com/certificates



Hybrid Professional Master's Degree in Software Development

General Structure of the Syllabus

Subject type	ECTS
Compulsory (CO)	60
Optional (OP)	0
External Work Placement (WP)	4
Master's Degree Thesis (MDT)	0
	Total 64

General Structure of the Syllabus

Year	Subject	ECTS	Type
1º	Programming Fundamentals	6	CO
1º	Data Structure	6	CO
1º	Algorithm and Complexity	6	CO
1º	Databases	6	CO
1º	Advanced Databases	6	CO
1º	Advanced Algorithms Design	6	CO
1º	Human-Computer Interaction	6	CO
1º	Advanced Programming	6	CO
1º	Development of Web Applications	6	CO
1º	Software Engineering	6	CO

[illegible]
Dr. Pedro Navarro Illana
Dean

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

tech global university

Hybrid Professional Master's Degree
Software Development

Modality: **Hybrid (Online + Internship)**
Duration: **12 months**
Certificate: **TECH Global University**
Accreditation: **60 + 4 ECTS credits**

Hybrid Professional Master's Degree
Software Development

tech global university