COMPUTER SYSTEMS ENGINEERING

Courses offered during Winter semester (academic year 2026-27)

Mathematical analysis (1st year, 6 ECTS)

At the end of this curricular unit, the student should be able to:

- Calculate derivatives and use this concept in optimization problems.
- Calculate partial derivatives, total derivatives and gradients of multivariable functions.
- Interpret the concepts of definite and indefinite integrals.
- Calculate integrals using integration formulas and applying advanced integration techniques.
- Identify and solve 1st order ordinary differential equations.
- Model real problems using 1st order ordinary differential equations.
- Identify the convergence or divergence of a series.
- Apply Taylor series to approximations.

Syllabus

1. Differentiation in R and Rn
1.1 Derivative function. Applications.
1.2 Partial Derivatives. Chain rule.
1.3 Gradients.
2. Integrals
2.1 Definite and Indefinite integrals.
2.2 Integrations techniques.
2.3 Fundamental Theorem of Calculus.
2.4 Application of definite integrals in the calculation of areas.
3. 1st Order Ordinary Differential Equations
3.1 Variable separation.
3.2 Linear differential equations.
3.3 Exact differential equations.
3.4 Applications of ODEs.
4. Series
4.1 Convergence criteria.
4.2 Taylor series.

Teaching methodology and assessment:

Since this curricular belongs to the scientific area of Mathematics, the teaching methodologies have been structured to ensure articulation with a pedagogical model that promotes:

- active and participative learning through the intensive use of classes to solve exercises and the proposal of exercises to be solved through autonomous study;
- continuous support and guidance to the students based on the monitoring of student's performance to identify difficulties;
- the use of computer tools to solve exercises, to stimulate students' interest in the subjects being studied.

Students' assessment in this subject is structured on a continuous basis and covers different components, aiming to ensure a comprehensive understanding of the concepts and techniques covered throughout the subject. The components of this assessment system are the following: 2 individual tests, group assignments and a comprehensive final exam.

Bibliography

[1] Faria, A.; Brás, H.; Figueiredo, I. (2024). Análise Matemática I. 2ª Edição. Edições Sílabo.
[2] Faria, A.; Brás, H.; Figueiredo, I. (2023). Análise Matemática II. 1ª Edição. Edições Sílabo.
[3] Anton, H.; Bivens, I; Davis, S. (2022) Calculus. 12th Edition. Wiley.
[4] Croft, A.; Davison, R.; Flint, J.; Hargreaves, M. (2017) Engineering Mathematics, 5th Edition. Pearson.

Software Engineering (1st year, 6 ECTS)

At the end of this curricular unit, the student should be able to:

- Apply the software engineering process
- Describe and choose development methodologies
- Apply project planning with a management tool using a given methodology
- Analyze, identify, and use UML diagrams
- Describe and apply software patterns
- Identify and produce different types of documentation
- Apply REST and MVC application architectures
- Use a Version Control System
- Configure and use logging, static code analysis, and optimization tools
- Apply and define automated tests, distinguish different types of tests and their contexts of use

Syllabus:

1. Introduction
1.1 Objectives, syllabus, and evaluation
1.2 Software engineering processes
2. Development methodologies
2.1 Classical methodologies
2.2 Agile methodologies
3. Software project management
3.1 Planning
3.2 Documentation
3.3 Development cycle
3.4 Risk management
4. Software engineering methods
4.1 Architectures and software models
4.2 Design patterns
5. Software engineering tools
5.1 Version control
5.2 Static analysis
5.3 Debugging and Profiling
5.4 Code optimization
6. Software engineering practice
6.1 Development tools
6.2 Automated testing
6.3 Continuous integration and deployment

Teaching methodology and assessment:
Assessment in the Software Engineering curricular unit is structured on a continuous basis and covers different components to ensure a comprehensive understanding of the concepts and techniques covered throughout the subject. Assessment includes two tests during the semester, the project and a compulsory comprehensive final exam.
Assessment Elements: 2 Tests, Project, Final Exam.

Bibliography:
[1] Gamma, E., Helm, R., Johnson, R., Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.
[2] Guerreiro, S. (2015). Introdução à Engenharia de Software, FCA.
[3] Borges, José; Dias, Teresa e Cunha, João (2015), Modelação de Dados em UML. Uma Abordagem por Problemas. Lisboa: FCA.
[4] Pressman, R., (2014). Software Engineering: A practitioner's Approach", 8th Ed., McGraw-Hill.
[5] Sommerville, I. (2015), Software Engineering. Pearson Education.
[6] IEEE Computer Society. (2024). Software Engineering Body of Knowledge (SWEBOK), Version 4.0. Disponível em https://www.computer.org/education/bodies-of-knowledge/software-engineering/

Applied Statistics (1st year, 6 ECTS)

At the end of this curricular unit, the student should be able to:

- Describe basic statistical concepts, such as population, sample, variables and measurement scales.
- Characterize and interpret data sets through frequency tables, graphs and calculation of measures.
- Evaluate the degree of linear association of bivariate data.
- Calculate probabilities and apply the concept of probability as a measure of uncertainty.
- Identify and apply theoretical models of probability distribution.
- Calculate and interpret confidence intervals about the parameters of a given population.
- Perform parametric hypothesis tests on the parameters of a given population, evaluating the significance of the results obtained.
- Use excel program to perform tables, graphs and calculations using statistical functions.

Syllabus:
1. Basic Concepts
1.1 Objectives
1.2 Key Concepts: Population, Sample, Variables, and Scales of Measurement
2. Descriptive Statistics
2.1 Sample Characterization: Tables and Graphical Representations
2.2 Central Tendency, Dispersion, Skewness, and Kurtosis
2.3 Correlation and Linear Regression Analysis
3. Probability Theory
3.1 Random Experiments, Sample Spaces, and Events
3.2 Conditional Probability and Bayes' Theorem
4. Random Variables and Probability Distributions
4.1 Discrete Probability Models (Poisson, Binomial, Hypergeometric)
4.2 Continuous Probability Models (Normal and Negative Exponential)
5. Confidence Interval Estimation
5.1 Estimation for Means and Mean Differences
5.2 Proportion Estimation and Differences in Proportions

5.3 Determining Sample Size
6. Hypothesis Testing
6.1 Procedure in hypothesis testing
6.2 Parametric Tests

Teaching methodology and assessment:
Continuous assessment with comprehensive examination with 3 individual assessment tests and a comprehensive exam.

Bibliography:
[1] André, J. (2018). Probabilidades e Estatística para Engenharia. Lidel.
[2] Montgomery, D.C., Runger, G. C. (2018) Applied Statistics and Probability for Engineers. Wiley.
[3] Pedrosa, A.C. and Gama, S.M. A. (2016) Introdução computacional à Probabilidade e Estatística com Excel. Porto Editora
[4] Reis, E.; Melo, P; Andrade, R e Calapez, T. (2021) Estatística Aplicada. Vol. 1 e 2. Edições Sílabo.

Programming Principles (1st year, 6 ECTS)

At the end of this curricular unit, the student should be able to:

- Formulate problems in natural language and their translation into an abstract language
- Explain the notion of algorithm, and master its description using various structured notations
- Explain and simulate the execution of an algorithm and its results
- Correctly distinguish and use programming concepts such as control structures, data types and declarations, operators and expressions
- Apply procedures and functions as a way of structuring algorithms
- Apply basic data structures such as arrays, and strings in algorithms
- Apply structured types and files to algorithms
- Apply procedural programming language C to implement introductory algorithms

Syllabus:
1. Introduction to Programming and Algorithms
1.1 Representation of Algorithms
1.2 Execution and Testing Algorithms
2. Types and Declarations of Data, Expressions and Intrinsic Functions
2.1 Data types: ordinal, real, string, definition and type conversion
2.2 Constants
2.3 Operators and expressions
2.4 Intrinsic functions
3. Control structures
3.1 Simple and structured instructions
3.2 Instructions: conditional, repetitive, jumping
3.3 Built-in structures
4. Functions
4.1 Concept
4.2 Local vs. Global Variables
4.3 Functions
4.4 Recursion
5. Indexed Variables and Strings
5.1 Concepts
5.2 Vectors and Matrices

5.3 Character Strings
6. Structured Types and Files
6.1 User-defined records
6.2 Notion, type and use of files
6.3 Text and direct access files

Teaching methodology and assessment:
The proposed methodology involves three main components:

- theoretical-practical classes,
- practical laboratory classes and
- integrated group programming projects.

Assessment in the Programming Fundamentals course is structured on a continuous basis and covers different components, with the aim of guaranteeing a comprehensive understanding of the concepts and techniques covered throughout the course.
Programming Principles is a project subject, which requires a positive assessment in this element to guarantee success. Assessment also includes two tests during the academic semester and a compulsory comprehensive final exam.
Assessment Elements: 2 Tests, Project, Final Exam

Bibliography:
[1] Marques de Sá, J. P. "Fundamentos de Programação Usando C", Editora FCA, 2004.
[2] Damas, Luis "Linguagem C", Editora FCA, 1999.
[3] Sampaio, Isabel; Sampaio, Alberto "Fundamental da Programação em C", Editora FCA, 1998.
[4] Kernighan, Brian W. "The C Programming Language", Prentice-Hall International, 1988.
[5] Cormen, Leiserson, Rivest "Introduction to Algorithms " MIT-Press, 1990.

Digital Systems (1st year, 6 ECTS)

At the end of this curricular unit, the student should be able to:

- Represent quantities in the binary, octal and hexadecimal systems and carry out conversions.
- State the postulates, properties and theorems of Boolean algebra and apply them to the design of DS.
- Describe the behaviour of the fundamental logic functions/logic gates AND, OR, NOT, NAND, NOR, XOR and NXOR.
- Identify the phases of the DS project.
- Simplify functions using systematic methods.
- Use SSI gates and MSI circuits to design complex DS.
- Implement physical digital circuits in the laboratory, using SSI/MSI (TTL/CMOS) components and integrated circuits.

Syllabus:
1. Principles of Digital Systems.
1.1 Characteristics
1.2 Analog and digital signals
1.3 Integration scales
2. Number systems.
2.1 Binary
2.2 Octal and Hexadecimal
2.3 Conversions
3. Logical functions and Boolean algebra.

3.1 Logic functions
3.2 Boolean Algebra
3.3 Canonical forms
3.4 Simplification methods.
4. Small Scale Integration circuits (SSI)
4.1 Logic gates.
4.2 Implementation with NAND or NOR gates
4.3 SSI chips characteristics.
4.4 Positive and negative logic.
5. Medium Scale Integration circuits (MSI): combinational and sequential
5.1 Coders and decoders
5.2 Multiplexers and demultiplexers
5.3 Comparators, converters and arithmetic circuits
5.4 Latches and Flip-Flops
6. Laboratory projects.
6.1 Variables and logic functions
6.2 SSI circuits - gates in TTL/CMOS chips
6.3 Combinatorial circuits - design with SSI gates
6.4 MSI circuits - BCD-7 segments decoder/multiplexers

Teaching methodology and assessment:
The teaching and learning methodology for the subject of Digital Systems is structured to promote a balanced combination of theory and practice, integrating different approaches to stimulate the active involvement of students and the application of acquired knowledge in a laboratory context. The proposed methodology involves two main components: theoretical-practical classes and practical laboratory classes.
This course has been designed to ensure coherence between the teaching and assessment methodologies and the learning objectives, providing solid training in the synthesis, minimization and implementation of digital systems.

Bibliography:
[1] Tocci, R., Widmer, N., Moss, G., (2021). Digital Systems: principles and applications, 12th ed., Pearson.
[2] Elahi, A. (2022). Computer Systems: Digital Design, Fundamentals of Computer Architecture and ARM Assembly Language, Springer.
[3] Floyd, T. L. (2021). Digital Fundamentals, 11th ed., Pearson.
[4] Baptista, C. P. (2015). Introdução aos Sistemas Digitais, FCA – Editora de Informática.
[5] Ribeiro, N. M. (2024). Protocolos para Trabalhos Laboratoriais de Sistemas Digitais, Faculdade de Ciência e Tecnologia, UFP. (online)

Algorithms and Data Structures (2nd year, 6 ECTS) prerequisites

At the end of this curricular unit, the student should be able to:

- Understand the notion of symbol tables and their applicability
- Master the notion of binary search trees and articulate associated algorithms
- Master the notion of tree balancing using red-black trees
- Distinguish the use of hash tables, hash functions and their properties
- Distinguish and apply the notion of graph in algorithmic modeling
- Master and use directed graphs in various algorithmic problems

- Apply minimum spanning tree algorithms and their properties
- Explain and apply shortest path calculation algorithms in graphs

Syllabus:
1. Introduction to non-linear data structures
1.1. Introduction to trees and graphs
1.2. Notation and Terminology
1.3. Representation
1.4. Applications of non-linear data structures
2. Search data structures
2.1. Symbol tables
3. Search and tree data structures
3.1. Introduction to Tree Data Structures
3.2. Binary search trees (BST - binary search trees)
3.3. Balanced trees
4. Hash tables
5. Unweighted Graphs
5.1. Introduction to the graph data structure and its representation
5.2. Graph traversal algorithm
5.3. Undirected graphs
5.4. Directed graphs
6. Weighted graphs
6.1. Weighted graphs
6.2. Weighted and directed graphs

Teaching methodology and assessment:
The proposed methodology involves three main components: theoretical-practical classes, practical laboratory classes and integrated group programming projects.
Assessment is structured on a continuous basis and covers different components, with the aim of guaranteeing a comprehensive understanding of the concepts and techniques covered throughout the course. The Algorithms and Data Structures course is a project course, which requires a positive assessment in this element to guarantee success in the course. Assessment also includes two tests during the academic semester and a compulsory comprehensive final exam.
Assessment Elements: 2 Tests, Project, Final Exam

Bibliography:
[1] Thomas H. Cormen, Leiserson, C., Rivest, R., & Stein, C. (2009). Introduction to Algorithms, third edition. MIT Press.
[2] Kleinberg, J., & Tardos, E. (2006). Algorithm Design. Pearson Education.
[3] Sedgewick, R., & Wayne, K. (2011). Algorithms, 4th Edition. Pearson Education.
[4] Vasconcelos, J., & Carvalho, J. V. de. (2005). Algoritmia e Estrutura de Dados: programação nas linguagens C e Java. Editora Centro Atlântico.
[5] Rocha, A. A. da. Estruturas de Dados e Algoritmos em Java. FCA - Editora de Informática, ISBN 978-972-722-704-4.
[6] Rocha, A. A. da. (2008). Estruturas de Dados e Algoritmos em C. FCA - Editora de Informática.

Human Computer Interaction (2nd year, 6 ECTS)

At the end of this curricular unit, the student should be able to:

- Identify the essential concepts of Human-Computer Interaction
- Analyze the principles associated with Human-Computer Interaction
- Describe the user-centered development cycle for digital systems
- Analyze the discovery of user needs as a phase of the user-centered development cycle
- Analyze design alternatives as a phase of the user-centered development cycle
- Analyze prototyping as a phase of the user-centered development cycle
- Analyze evaluation as a phase of the user-centered development cycle

Syllabus:

1. Human-Computer Interaction (HCI)
1.1. Basic concepts of HCI, ID, and UX
1.2. User-system relationship cycle
1.3. Historical evolution and work areas
2. Mediation Concepts in Digital Systems
2.1. Mediation
2.2. Human factors
2.3. Artifact, affordance, and mental models
3. HCI Development Cycle
3.1. Activity analysis
3.2. Principles of digital system design
3.3. User-centered development and lifecycle
4. Discovering User Needs
4.1. Ethics and best practices
4.2. Methods for gathering needs
4.3. Data and human task models
5. Design Alternatives
5.1. Crafting design alternatives
5.2. Creativity exercises
5.3. Personas, scenarios, and user models
6. Prototyping and HCI
6.1. Prototyping process
6.2. Low, medium, and high-fidelity prototypes
6.3. User feedback gathering
7. Evaluation in HCI
7.1. Project evaluation
7.2. Types of evaluation
7.3. Results processing

Teaching methodology and assessment:

The teaching and learning methodology is structured to promote a balanced combination of theory and practice, integrating different approaches to stimulate active student engagement and the application of acquired knowledge in real-world scenarios. The proposed methodology involves three main components: theoretical-practical classes, practical laboratory classes, and integrated group projects.

Assessment is structured on a continuous basis and covers different components to ensure a comprehensive understanding of the concepts and techniques covered throughout the course.

Assessment includes two tests during the semester, the project and a compulsory comprehensive final exam.

Bibliography:

[1] Sharp, Helen; Rogers, Yvonne and Preece, Jennifer (2019). Interaction Design: Beyond Human-Computer Interaction. 5ª Edition. Wiley.
[2] Gouveia, L. (2024). Notas sobre o tema de Interação Humano Computador. Reprografia da UFP.
[3] MacKenzie, I. (2013). Human Computer Interaction. An Empirical Research Perspective. Morgan Kaufmann. Elsevier.
[4] Shneidermam, Ben; Plaisant, Catherine; Cohen, Maxin and Jacobs, Steven. (2009). Designing the User Interface: Strategies for Effective Human-Computer Interaction. 5th Edition. Pearson.
[5] Norman, Donald. (2002). The Design of Everyday Things. Reprint edition. Basic Books.
[6] Krug, Steve (2015). Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability 3th Edition. Things that make us smart. Addison Wiley. Pearson Education.

Operational Research (2nd year, 6 ECTS) prerequisites

At the end of this curricular unit, the student should be able to:

- Formulate and interpret Linear Programming models.
- Solve Linear Programming problems graphically.
- Apply the Simplex Method to solve Linear Programming problems.
- Understand the concept of duality in Linear Programming.
- Solve problems using Integer Programming.
- Perform sensitivity analysis on Linear Programming models.

Syllabus:

1. Introduction to Operations Research: origin, objectives, and methodology.
2. Formulation of Linear Programming models.
3. Graphical solution of Linear Programming problems.
4. Simplex Method: fundamentals and application.
5. Duality in Linear Programming.
6. Integer Programming: formulation and solution methods.
7. Sensitivity analysis in Linear Programming models.

Teaching methodology and assessment:

This curricular unit aims to provide students with a practical and applied understanding of Operations Research concepts, emphasizing the formulation and resolution of optimization problems. Through an approach centered on applying theoretical knowledge to concrete scenarios, students will develop analytical and practical skills to interpret and solve complex problems using modern and adaptive tools.
The assessment is structured as a continuous process and encompasses different components, aiming to ensure a comprehensive understanding of the concepts and techniques covered throughout the curriculum. The assessment methodology consists of two assessments, each contributing 50% to the continuous assessment.

Bibliography:

[1] Hill, M. M., & dos Santos, M. M. (2022). Investigação Operacional–Vol. 1–Programação Linear.
[2] Hill, M. M., & dos Santos, M. M. (2018). Investigação Operacional–Vol. 2–Exercícios de Programação Linear.
[3] de Jesus, F., & Lisboa, J. V. (2020). Introdução à investigação operacional. Vida Económica Editorial

[4] Mourão, M. C., Pato, M. V., Pinto, L. S., L., Simões, O. A., Valente, J. (2019). Investigação Operacional: Exercícios e Aplicações. Escolar Editora.

Object-Oriented Programming (2nd year, 6 ECTS) prerequisites

At the end of this curricular unit, the student should be able to:

- List and describe the main concepts of OOP
- Analyze and draw UML class diagrams
- Use data types, operators and expressions, flow control and data collections to implement UML class diagrams
- Identify and interpret errors in the compilation and execution of OO programs in Java
- Apply and implement classes for developing OO applications using Java
- Design and implement graphical user interfaces (GUI) using packages
- Properly structure and handle the persistent storage of information
- Create and use data structures using generics

Syllabus:

1. Introduction to Object-Oriented Programming
1.1 Designing OO software
1.2 Classes versus Objects
1.3 Messages, inheritance and polymorphism
2. CASE tools
2.1 UML methodologies
2.2 Class diagrams
2.3 Class implementation
3. Introduction to Java
3.1 Classes, attributes, data types and encapsulation
3.2 Classes, methods, operators and flow control
3.3 Polymorphism (overload and override)
3.4 Arrays and collections
3.5 Contracts (interfaces)
3.6 Inheritance
4. Runtime problems
4.1 Errors
4.2 Exceptions
4.3 Definition of exceptions
5. Files
5.1 Handling the file system
5.2 Text and binary files
5.3 Random access files
6. Advanced concepts
6.1 Generics
6.2 Functional programming
7. Graphical interfaces
7.1 Hierarchy of graphic components
7.2 Containers and layout managers
7.3 Graphical components
7.4 Event-based programming

Teaching methodology and assessment:
The Object-Oriented Programming (OOP) course adopts a pedagogical model that combines theory and practice, providing students with an in-depth understanding of the fundamental concepts of OOP applied in Java. This approach is based on theoretical-practical and laboratory classes, focusing on the practical and incremental application of skills, in line with the learning objectives (OA).
The assessment is designed to provide a comprehensive measure of the students' progress in acquiring essential OOP skills, as well as in using the Java language to implement class diagrams and OO solutions.

Bibliography:
[1] Eckel, Bruce, Thinking in Java, 4th Ed., Prentice Hall, 2006.
[2] Paul Deitel, Harvey Deitel, Java How to Program, Early Objects, 11th Ed., 2018.
[3] W3Schools, Java OOP. URL: https://www.w3schools.com/java/java_oop.asp (Last visited: Nov 2024)

Operating Systems (2nd year, 6 ECTS) prerequisites

At the end of this curricular unit, the student should be able to:

- Describe the structure and functioning of the operating system.
- Compare processes and threads. States, transitions, context switches.
- Analyse different process and thread scheduling algorithms.
- Program multi-process and multi-threaded applications.
- Analyse and program different inter-process communication mechanisms.
- Describe and apply synchronisation mechanisms in concurrent applications based on classic process and thread coordination problems.
- Describe memory management mechanisms in a modern operating system
- Describe the organisation and operation of the most important file systems.
- Describe the main input/output mechanisms, peripheral classes and device drivers.

Syllabus:
1. Introduction to Operating Systems
1.1 Operating system software design and approaches
1.2 Abstractions, processes, and resources
1.3 Executing a System Call
2. Processes and Threads
2.1 Process and thread concept and life cycle
2.2 Inter-process communication
2.3 Process and thread scheduling algorithms
3. Concurrency PC 3.1 – Mutexes, Semaphores, Condition Variables
3.2 Classical synchronization problems
4. Memory Management
4.1 Linear memory, swapping, virtual memory
4.2 Page replacement algorithms
5. File Systems
5.1 Files, Directories,
5.2 File Systems design and structure
6. Device Management
6.1 Buffering Strategies
6.2 Device drivers

Teaching methodology and assessment:
The teaching/learning methodology is structured to promote a balanced combination of theory and practice, integrating different approaches to stimulate students' active involvement and the application of acquired knowledge in real-life scenarios. The proposed methodology involves three main components: theoretical-practical classes, practical laboratory classes and integrated group programming projects.
Assessment is structured on a continuous basis and covers different components, with the aim of guaranteeing a comprehensive understanding of the concepts and techniques covered throughout the course. The operating systems course is a project course, which requires a positive assessment in this element to guarantee success in the course. Assessment also includes 2 tests during the academic semester and a compulsory comprehensive final exam.

Bibliography:
[1] Operating System Concepts" de Abraham Silberschatz, Peter B. Galvin e Greg Gagne, 10th ed., Wiley 2021
[2] "Modern Operating Systems" de Andrew S. Tanenbaum e Herbert Bos , 5th ed, Pearson 2023
[3] "Operating Systems: Three Easy Pieces" de Remzi H. Arpaci-Dusseau e Andrea C. Arpaci-Dusseau , online
[4] "Unix and Linux System Administration Handbook" de Evi Nemeth, Garth Snyder, Trent R. Hein, Ben Whaley e Dan Mackin, 5th ed., Addison-Wesley 2017

Multimedia I (3rd year, 6 ECTS) prerequisites

At the end of this curricular unit, the student should be able to:

- Identify and contextualize the concept of multimedia, the different types of multimedia information, their characteristics and associated technologies
- Understand the digital representation of information by distinguishing analog from digital signals
- Recognize the digitization process and the specificities of interactivity
- Classify and distinguish interactive multimedia applications
- Understand the different types of static and dynamic media
- Understand the general concepts related to the field of computer graphics
- Understand the necessary bases for the construction of interactive graphic systems
- Develop interactive graphics systems

Syllabus:
1. Introduction to multimedia
1.1. Contextualization of the multimedia concept
1.3. Types of multimedia information
1.4. Definition of multimedia
1.5. Characterization of multimedia systems
1.6. Multimedia technologies
2. Digital information and interactivity
2.1. Digital representation of information
2.2. Notion of interactivity
3. Types of multimedia information
3.1. Static media
3.2. Dynamic media
4. Interactive Multimedia Applications
4.1. Classification of interactive multimedia applications

4.2. Types of interactive multimedia applications
5. Computer Graphics
5.1. Introduction to computer graphics
5.2. History of computer graphics
6. Development of graphics systems
6.1. Introduction to OpenGL
6.2. Structure of graphic programs
6.3. Points, Lines and Polygons
6.4. Geometric objects
6.5. Display Lists and Animations
6.6. Geometric transformations
6.7. Projections and models
6.8. Lighting and textures

Teaching methodology and assessment:
The transmission of knowledge in this course will be carried out using theoretical-practical and practical classes in a laboratory environment. Theoretical-practical classes present the fundamental concepts for understanding the themes of the program. In practical laboratory classes, students are confronted with exercises that they have to solve using tools for developing graphic systems based on OpenGL. The assessment is divided into two components: theoretical (50%) and practical (50%). In the first case, the evaluation results from a written test applied during the semester on the subject taught in the classroom. In the second case, it results from a practical work that consists in the development of a graphics project that is carried out and defended by the students throughout the semester.

Bibliography:
[1] Ribeiro, N. M., “Multimédia e Tecnologias Interativas”, FCA, 2012.
[2] Chapman, N., Chapman, J., “Digital Multimedia”, 3rd Edition, John Wiley and Sons, 2009.
[3] Pereira, J. M., Brisson, J., Coelho, A., Ferreira, A., Gomes, M. R., “Introdução à Computação Gráfica”, FCA, 2018.
[4] Foley, J. D., van Dam, A., Feiner, S. K., Hugues, J. F., Phillips, R. L., "Introduction to Computer Graphics", Addison-Wesley, 1993.
[5] Hughes, J. F., van Dam, A., McGuire, M., Sklar, D. F., Foley, J. D., Feiner, S. K., Akeley, K., “Computer Graphics: Principles and Practice”, 3rd Edition, Addison-Wesley, 2013.
[6] Shreiner, D., Woo, M., Neider, J., Davis, T., "OpenGL Programming Guide: The Official Guide to Learning OpenGL", Addison-Wesley, 2013.

Programming Laboratory (3rd year, 5 ECTS) prerequisites

At the end of this curricular unit, the student should be able to:

- Design and plan a Web application.
- Elaborate a Mockup of a Web application.
- Understand and describe the principles of full-stack, server-side and client-side programming.
- Know the MVC architecture organization.
- Able to develop a Web application and/or Rest API.
- Able to work and develop views that will then be interpreted by the template engines.
- Able to develop responsive Web pages using existing responsive CSS/JS frameworks.

Syllabus:

0. Presentation
0.1 Background and Goals
0.2 Methodology and Evaluation System
0.3 Syllabus and Bibliography
1. Introduction HTML / CSS
1.1 Markup language
1.2 Responsive Framework
2. Introduction to PHP Language
2.1 Data Types
2.2 Operators
2.3 Control Structures
2.4 Repetition Structures
2.5 Functions
3. PHP Programming Syntax
3.1 Exception Handling
3.2 File Usage (include, require)
3.3 Object Oriented Programming (OO) Fundamentals
3.3.1 Classes / Instances
3.3.2 Constructors / Destructors
3.3.3 Encapsulation
3.3.4 Inheritance
3.3.5 Interfaces
3.3.6 Abstract classes and methods
4. Web Applications
4.1 HTML forms
4.2 Requests
4.3 Sessions
4.4 Interaction with a database - PDO
5. MVC Web Development Framework
5.1 Routes
5.2 Controllers
5.3 Views
5.4 ORM

Teaching methodology and assessment:

The knowledge transmission will be achieved by explaining practical exercises during the practical classes. In these classes' students are faced with real problems that they have to solve. The final grade will be calculated according to the following formula:
Assessment: test (55%), project (35%), worksheets (10%)

Bibliography:

[1] PHP com programação orientada a objetos / Frederico Tavares, FCA, 2016.
[2] Introdução ao desenvolvimento moderno para a web - do front-end ao back-end: uma visão global! / Filipe Portela, FCA, 2018.
[3] Typescript - o javascript moderno para criação de aplicações / Luís Abreu, FCA, 2017.
[4] Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5 (Learning PHP, MYSQL, Javascript, CSS & HTML5) / Robin Nixon, 5ª ed., O'Reilly Media, 2018.
[5] HTML, CSS & JavaScript Web Publishing in One Hour a Day, Sams Teach Yourself: Covering HTML5, CSS3, and jQuery / Laura Lemay, 7ª ed., Sams Publishing, 2016.

[6] PHP com Programação Orientada a Objetos / Frederico Tavares, FCA, 2016.

Courses offered during Spring semester (academic year 2026-27)

Linear Algebra (1st year, 6 ECTS)

At the end of this curricular unit, the student should be able to:

- Calculate sums and products of matrices.
- Relate a matrix to a system of linear equations, and use operations with matrices to obtain the solution of this system.
- Calculate determinants using different methods.
- Test the linear independence of a set o vectors.
- Find a basis for a vector space.
- Calculate the coordinates of a vector in different bases.
- Orthogonalize vectors.
- Calculate eigenvalues and eigenvectors of a matrix and diagonalize the matrix.

Syllabus:

1. Matrices
1.1 Operations with matrices.
1.2 Matrices and linear equations systems.
2. Determinants
2.1 Laplace's theorem.
2.2 Properties of determinants.
3. Vector Spaces
3.1 Linear independence.
3.2 Basis and dimension.
4. Linear Transformations
4.1 Change of basis and orthogonalization.
4.2 Eigenvectors and eigenvalues.

Teaching methodology and assessment:

Since this CU belongs to the scientific area of Mathematics, the teaching methodologies have been structured to ensure articulation with a pedagogical model that promotes:

- active and participative learning through the intensive use of classes to solve exercises and the proposal of exercises to be solved through autonomous study;
- continuous support and guidance to the students based on the monitoring of student's performance to identify difficulties;
- the use of computer tools to solve exercises, to stimulate students' interest in the subjects being studied.

Students' assessment in the course "Linear Algebra" is structured on a continuous basis and covers different components, aiming to ensure a comprehensive understanding of the concepts and techniques covered throughout the course.
The components of this assessment system are the following: 2 individual tests, group assignments and comprehensive final exam.

Bibliography:

[1] Cabral, I.; Perdigão, C.; Saiago, C. (2024). Álgebra Linear: Teoria, Exercícios Resolvidos e Exercícios Propostos com Soluções. 7ª Edição. Escolar Editora.
[2] Gonçalves, R. (2022). Álgebra Linear – Teoria e Prática. 3ª Edição. Edições Sílabo.
[3] Thulasidas, M. (2021). Linear Algebra for Computer Science. Asian Books, Singapure.
[4] Anton, H.; Kaul, A. (2019). Elementary Linear Algebra. 12th Edition, Wiley.

Algorithm Analysis and Design (1st year, 6 ECTS) prerequisites

At the end of this curricular unit, the student should be able to:

- Describe and use the main data structures used in algorithms
- Explain and analyze algorithmic complexity
- Explain the importance of algorithm design and its impact on performance
- Apply principles of algorithmic efficiency in particular cases
- Apply search techniques to strings like KMP
- Identify and use elementary sorting methods
- Identify and use sort methods such as merge sort and quick sort
- Identify and use abstract data types such as stacks, queues, and priority queues

Syllabus:

1. Fundaments of Algorithms and Data Structures 1.1. Algorithm representation and programming models
1.2. Introduction to linear and nonlinear data structures
1.3. Analysis of algorithmic complexity
1.4. Introduction to Algorithmic Design Techniques
2. Applications and case studies of Analysis of algorithmic complexity
3. Text processing algorithms (Strings)
3.1. Introduction to operations with strings
3.2. Sorting operations
3.3. Search operations
4. Elementary Data Structures
4.1. Abstract Data Types (ADT)
4.2. Arrays
4.3. Linked Lists
4.4. Stacks
4.5. Queues
5. Sorting
5.1. Introduction to the sorting problem
5.2. Elementary sorting methods
5.3. Merge Sort
5.4. QuickSort
6. Heaps, priority queues and Heap Sort

Teaching methodology and assessment:

This curricular unit is structured on a continuous basis and covers different components, with the aim of guaranteeing a comprehensive understanding of the concepts and techniques covered throughout the course.
This curricular unit is a project, which requires a positive assessment in this element to guarantee success in the subject. Assessment also includes 2 tests during the academic semester and a compulsory comprehensive final exam.

Bibliography:

[1] Thomas H. Cormen, Leiserson, C., Rivest, R., & Stein, C. (2009). Introduction to Algorithms, third edition. MIT Press.
[2] Kleinberg, J., & Tardos, E. (2006). Algorithm Design. Pearson Education.
[3] Sedgewick, R., & Wayne, K. (2011). Algorithms, 4th Edition. Pearson Education.
[4] Vasconcelos, J., & Carvalho, J. V. de. (2005). Algoritmia e Estrutura de Dados: programação nas linguagens C e Java. Editora Centro Atlântico.
[5] Rocha, A. A. da. Estruturas de Dados e Algoritmos em Java. FCA - Editora de Informática, ISBN 978-972-722-704-4.
[6] Rocha, A. A. da. (2008). Estruturas de Dados e Algoritmos em C. FCA - Editora de Informática.

Computer Architecture (1st year, 6 ECTS) prerequisites

At the end of this curricular unit, the student should be able to:

- Interpret basic concepts and technology underlying computer architecture, including numeric representation.
- Explain the function of the instruction set of an architecture, focusing on MIPS, and perform memory and control flow operations.
- Apply IEEE 754 standards for floating-point operations and their implementation in MIPS.
- Identify and describe processor components and the instruction cycle, including control and data flow.
- Analyze and solve issues related to pipelining and its application in MIPS architecture.
- Explain memory hierarchy and perform performance analyses on cache systems.
- Demonstrate knowledge of input/output mechanisms and implement methods for programmed I/O and interrupt-driven I/O.

Syllabus:

1. Introduction to Computer Architecture
1.1 Basic concepts and technology
1.2 Integer representation and arithmetic operations
2. Instruction Set
2.1 MIPS architecture
2.2 Memory access, decision-making, and procedures
2.3 Instruction representation and translation
3. Floating-Point Representation
3.1 IEEE 754 standards and arithmetic operations
3.2 Implementation in MIPS
4. Processor: Data and Control
4.1 Basic implementation of MIPS
4.2 Instruction cycle and hardware
5: Introduction to Pipelining
5.1 Concepts and pipelining issues
5.2 Pipelining in MIPS
6. Memory Hierarchy
6.1 Cache systems and performance analysis
7. Input/Output Mechanisms
7.1 Programmed I/O and interrupt-driven I/O
7.2 I/O support in MIPS

Teaching methodology and assessment:

- Theoretical and theoretical-practical classes for the presentation and discussion of fundamental concepts, promoting understanding and critical analysis.
- Practical laboratory sessions, where students solve real-world problems using simulators like Mars to apply the concepts.
- Practical exercises and projects that allow consolidation of knowledge on MIPS architecture and programming at the hardware level.

Assessment: 2 tests, final project, comprehensive exam

Bibliography:

[1]Patterson, D., & Hennessy, J. – Computer Organization and Design: The Hardware/Software Interface, 5ª ed., Morgan Kaufmann, 2020.
[2] Tanenbaum, A. S. – Structured Computer Organization, 6ª ed., Prentice Hall, 2013.
[3] Stallings, W. – Computer Organization and Architecture, 10ª ed., Prentice Hall, 2015.

Physics (1st year, 6 ECTS)

At the end of this curricular unit, the student should be able to:

- Identify and convert used unit systems into physics and perform vector operations.
- Identify different types of forces and perform composition and resolution of these forces.
- Solve static problems that involve the application of a particle's equilibrium conditions.
- Describe rectilinear motion and relate forces to acceleration.
- Calculate the work done and the energy associated with the motion of a particle.
- Apply Coulomb's law to calculate the electric force between point charges.
- Examine the direction and strength of electric fields produced by different charge distributions.
- Recognize the nature of magnetism and its consequences.
- Calculate and interpret the Lorentz force on charged particles moving in a magnetic field.
- Perform laboratory experiments and communicate and critically evaluate the results.

Syllabus:

1. Introduction to Mechanics
1.1. Unit systems.
1.2. Scalars, vectors and vector operations.
1.3. Fundamental concepts and principles.
2. Particle Equilibrium
2.1. Types of forces.
2.2. Resulting from a system of forces.
2.3. Conditions of equilibrium of a particle in the plane.
3. Particle dynamics
3.1. Kinematics: rectilinear motion
3.2 Application of Newton's 2nd law
3.3. Work and Energy
4. Electric Field
4.1. Electrical charge. Electric force and Coulomb's law.
4.2. Electric field and field lines. Electrical load distributions.
4.3. Electric potential. Notion of gradient. Equipotential surface. Electrical Potential of a distribution of electrical charges.
4.4. Electric field as a gradient of Electric potential.
5. Magnetic field

5.1. Magnetic field and Oersted experiment. Biot-Savart's Law.
5.2. Lorentz force.

Teaching methodology and assessment:
The pedagogical model of this course unit is based on the practical application of theoretical concepts, promoting active and participatory learning that encourages student responsibility and autonomy.
Continuous assessment with comprehensive examination: continuous assessment includes elements of continuous evaluation consisting of 2 tests and 2 laboratory worksheets and an alternative element that can replace any of the tests according to the one that is most favorable to the student. The alternative element consists of worksheets with exercises proposed in class or extra-class.

Bibliography:
[1] Beer, F.P. e Johnston, E.R. (2010) Mecânica Vetorial para Engenheiros: Estática, McGraw-Hill.
[2] Beer, F.P. e Johnston, E.R. (2012) Mecânica Vetorial para Engenheiros: Dinâmica, McGraw-Hill.
[3] Meriam, J.L. e Kraige, L.G. (2015) Mecânica - Estática, LTC.
[4] Meriam, J.L. e Kraige, L.G. (2022) Mecânica - Dinâmica, LTC.
[5] Halliday, D. Resnick, R. and Walker, J. (2018), Fundamentals of Physics, John Wiley & Sons.

Imperative programming (1st year, 6 ECTS) prerequisites

At the end of this curricular unit, the student should be able to:

- Identify and interpret compiling and executing errors in C programs
- Correctly use data types, operators and expressions, flow control, vectors, data structures, pointers and IO operations
- Apply functions in the design and implementation of C applications
- Define and implement dynamic data structures using pointers and structures
- Properly structure the storage of information in text and binary format
- Apply text and binary files to persistent information storage and retrieval
- Apply dynamic structures and files to solve concrete problems

Syllabus:
1. Introduction to C
1.1 Program structure
1.2 Variables and data types
1.3 Operators, expressions and flow control
2. Structuring programs in functions
2.1 Parameters and local variables
2.2 Recursion
3. Vectors and strings
3.1 Declaration and initialisation
3.2 Function parameters
3.3 Multi-dimensional vectors
4. Pointers
4.1 Declaration and initialisation
4.2. Pointer arithmetic
4.3 Reference passing
4.4 Pointers to pointers
5. Data structures
5.1 Declaration of types

5.2 Using structures
6. Pointers and dynamic data structures
6.1 Dynamic vectors with structures
6.2 Managing linked Stacks and Queues
7. Files
7.1 Sequential text and binary files
7.2 Random access files
8. Advanced concepts
8.1 Preprocessor instructions
8.2 Register, enumeration and union variables
8.3 Functions and static variables
8.4 Pointers to functions

Teaching methodology and assessment:
This curricular unit adopts a teaching methodology based on theoretical-practical and laboratory classes, focusing on the practical application of fundamental C programming concepts and the incremental development of students' technical skills.
The assessment will consist of: 2 tests, project, comprehensive exam

Bibliography:
[1] L. Damas, Linguagem C, 10ª Edição, FCA, 1999
[2] Brian W. Kernighan, Dennis Ritchie, The C Programming Language, 2nd Edition, Pearson, 1988
[3] Robert C. Seacord, Effective C: An Introduction to Professional C Programming, No Starch Press, 2020
[4] GeeksForGeeks, C Programming Language Tutorial, URL: https://www.geeksforgeeks.org/c-programming-language (Last visited 2024)

Data Analysis and Visualization (2nd year, 6 ECTS) prerequisites

At the end of this curricular unit, the student should be able to:

- Describe the importance of data collection and preparation for effective analysis.
- Learn and apply preprocessing techniques to prepare data for analysis.
- Develop skills to perform exploratory analysis, understanding patterns, trends, and anomalies in data.
- Learn to create visualizations that communicate insights clearly and effectively. OA5 Consolidate knowledge through integrated practice, applying all stages of data analysis.

Syllabus:
1. Introduction to Data Preparation and Import
1.1. Data import (CSV, SQL, APIs).
1.2. Data types and structures.
1.3. Basic cleaning and standardization.
1.4. Observation unit and analytical scope.
2. Data Preprocessing and Transformation
2.1. Normalization, standardization, and encoding.
2.2. Handling missing data.
2.3. Dimensionality reduction.
2.4. Variable creation (feature engineering).
3. Exploratory Data Analysis (EDA)
3.1. Descriptive statistics and data distribution.

3.2. Outlier treatment.
3.3. Missing data analysis and impact.
3.4. Correlation between variables.
4. Data Visualization
4.1. Principles and graph selection.
4.2. Categorical and numerical visualizations.
4.3. Comparisons and temporal variations.
4.4. Popular tools.
5. Integrated Practice
5.1. Complete analysis.
5.2. Documentation and visualization.
5.3. Insights communication.

Teaching methodology and assessment:
The pedagogical model of this course unit is organized into three main pillars: active learning, practical integration, and continuous and collaborative feedback, structured to achieve the defined learning objectives:

- theoretical classes and interactive discussion
- practical laboratory sessions
- integrated practical project
- case studies and critical analysis
- continuous and competency-based assessment

Assessment criteria:

- integrated practical project
- final exam
- practical laboratory exercises
- participation and contribution in class

Bibliography:

[1] Nussbaumer Knaflic, C. (2017). Storytelling com dados: Um guia sobre visualização de dados para profissionais de negócios. Alta Books.
[2] Provost, F., & Fawcett, T. (2016). Data science para negócios. Alta Books.
[3] Wickham, H., & Grolemund, G. (2017). R para data science: Importe, arrume, transforme, visualize e modele dados. Alta Books.
[4] McKinney, W. (2018). Python para análise de dados. Novatec.
[5] Tukey, J. W. (1977). Análise exploratória de dados. Addison-Wesley.
[6] Alberto Cairo (2016). The Truthful Art: Data, Charts, and Maps for Communication. New Riders Publishing, USA.

Databases (2nd year, 6 ECTS)

At the end of this curricular unit, the student should be able to:

- Apply functional dependencies to normalize models expressed as sets of attributes.
- Apply models in a given dbms from requirements or a conceptual model.
- Apply relational algebra as a query language, given a query in natural language.
- Analyze and write sql queries given requirements in natural language.
- Analyze and justify the creation of indexes given a query.
- Analyze and explain concurrency control.
- Evaluate transactions suitable for a set of operations.

Syllabus:
1. The Relational Model of Data
1.1 Relations, constraints
1.2 Functional dependencies
1.3 Normalization, normal forms
2. Relational Algebra
2.1 Operators. Rules
2.2 Queries
2.3. Heuristic optimization
3. SQL
3.1. Relational Algebra and SQL
3.2 Schema definition. Constraints
3.3 Data insertion, modification and querying
3.4 Joins. Grouping and aggregate functions
3.5 Sub-selects and predicates
3.6 Set operations
3.7 Views
4. Indexes
4.1. How indexes work
4.2. Types of indexes
4.3. Using indexes
5. Concurrency control
5.1 Transactions. Schedules, conflicts
5.2 Serialization
5.3 Concurrency control algorithms. Locking
5.4 Isolation levels.

Teaching methodology and assessment:
The teaching and learning methodology for the Databases course is structured to promote a balanced combination of theory and practice, integrating different approaches to stimulate active student engagement and the application of acquired knowledge in real-world scenarios. The proposed methodology involves three main components: theoretical-practical classes, practical laboratory classes, and integrated group projects.
Assessment Elements: 2 Tests, Project, Final Exam

Bibliography:
[1]. Gouveia, Feliz (2021). Fundamentos de Bases de Dados, 2ª ed., Lisboa: FCA.
[2]. Gouveia, Luis (2024). Apontamentos selecionados sobre bases de dados. Reprografia da UFP.
[3]. Date, Chris J. (2003). An Introduction to Database Systems, 8º ed., Addison-Wesley, USA.
[4]. Luís Damas, SQL: Structured Query Language, 14ª edição. Lisboa: FCA.
[5]. Ullman, Jeffrey e Wisdom, Jeffrey. (2014), A First Course in database Systems, Prentice-Hall.

Hardware and Sensors (2nd year, 6 ECTS) prerequisites

At the end of this curricular unit, the student should be able to:

- Explore the application of sensors for monitoring and control in various domains.
- Describe the different types of existing sensors.
- Identify the technical development, installation and management challenges involved in designing sensory solutions.

- Describe the principles and technologies underlying the operation of the various types of sensors.
- Characterize the technologies that have influenced the evolution of intelligent sensory systems.
- Use the different platforms and components needed to integrate and collect information from sensors.
- Master the various ways of processing and acting on information read from sensors.

Syllabus:

1. Sensor fundamentals
1.1 Fundamental concepts of hardware and sensors
1.2 Microcontrollers
1.3 Application areas for sensors
2. Sensor systems
2.1 Mechanical sensors
2.2 Optical sensor
2.3 Semiconductor sensor
2.4 Electrochemical sensor
2.5 Biosensors
3. Main components of sensor technology
3.1 Embedded hardware and platforms
3.2 Microcontrollers for smart sensors
3.3 Embedded interfaces and communications
3.4 Sensor communications
4. Sensor network topologies and design
4.1 Sensor network components
4.2 Sensor network topologies
4.3 Sensor network applications
5. Sensor data processing
5.1 IoT - Internet of Things
5.2 Sensors and the cloud
5.3 Data extraction
5.4 Data visualization
6. Implementing projects with Arduino
6.1 Counter
6.2 Traffic lights
6.3 Intensity scale
6.4 Safe panel
6.5 Lift control
6.6 Morse code sending and receiving
6.7 Calculator
6.8. Building automation

Teaching methodology and assessment:

The proposed methodology involves two main components: theoretical-practical classes and practical laboratory classes.
The assessment includes 2 tests carried out during the semester, the reports corresponding to the experimental set-up of the laboratory protocols and a compulsory comprehensive final exam.

Bibliography:

[1] Monk, S. (2022). Programming Arduino: Getting Started with Sketches, 3rd ed., Tab Books.
[2] Thorpe, E. (2020). Arduino: Advanced Methods and Strategies of Using Arduino.
[3] Fraden, J. (2016). Handbook of Modern Sensors: Physics, Designs, and Applications, Springer.
[4] Karvinen,T., Karvinen,K., Valtokari, V. (2015). Make: Sensors: A Hands-On Primer for Monitoring the Real World with Arduino and Raspberry Pi, Make Community, LLC.
[5] Karvinen,T., Karvinen,K. (2014). Getting Started with Sensors: Measure the World with Electronics, Arduino, and Raspberry Pi, Make Community, LLC.
[6] Singh, J.(2023).Future Care: Sensors, Artificial Intelligence, and the Reinvention of Medicine, Mayo Clinic Press.

Client-Side Web Programming (2nd year, 6 ECTS) prerequisites

At the end of this curricular unit, the student should be able to:

- Interpret the basic structure and main elements of a web page by applying knowledge of HTML and CSS.
- Explain the syntax and main control structures of the JavaScript language.
- Develop and implement interactive functionalities in web pages using JavaScript.
- Apply concepts of web application development, including event handling and state management.
- Develop reusable components using the React framework to create dynamic user interfaces.
- Integrate HTML, CSS, JS and React into a practical client web application project

Syllabus:
1. Introduction to HTML/CSS
1.1 Structure and components of web pages
1.2 Styling and design with CSS
2. Introduction to JavaScript Language
2.1 History and characteristics of JavaScript
2.2 Introduction to client-side execution
3. JavaScript Syntax and Programming Structures
3.1 Control structures and data manipulation
3.2 Functions and DOM manipulation
4. Web Application Development
4.1 Event handling and interactivity
4.2 Storage and state management
5. React Framework
5.1 Basics of React
5.2 Components, props, and state

Teaching methodology and assessment:
Theoretical-practical classes for the presentation of HTML, CSS, JavaScript, and React fundamentals, aligning theory with practice across all CCs.
Laboratory sessions where students solve real-world web programming problems, developing pages and applications with immediate feedback.
Continuous mini-assignments and a Final Project, where students apply and integrate knowledge practically, consolidating skills in web development.
Assessment Elements: 2 tests, project, continuous assessment

Bibliography:
[1] Crockford, D. – JavaScript: The Good Parts. O'Reilly Media.

[2] Stefanov, S. – JavaScript Patterns. O'Reilly Media.
[3] Flanagan, D. – JavaScript: The Definitive Guide, 6th ed., O'Reilly Media.
[4] Haverbeke, M. – Eloquent JavaScript: A Modern Introduction to Programming, 2nd ed.
[5] ECMAScript® 2016 Language Specification – The JavaScript Standard Document.

Computer Networks (2nd year, 6 ECTS) prerequisites

At the end of this curricular unit, the student should be able to:

- List and define relevant terminology and equipment used in the context of computer networks.
- Describe the organisation of the Internet.
- Describe the layered structure and list the corresponding requirements of a typical network architecture.
- Define the principles of naming, addressing and locating resources.
- Describe the details of an application layer protocol.
- Describe the operation of reliable and unreliable transport protocols.
- Design, configure and test addressing on IP networks
- Analyse, configure and test routing protocols on IP networks
- Describe the characteristics of local area networks (LANs) and a media access protocol.
- Design and implement large-scale Ethernet networks

Syllabus:

1. Introduction to Computer Networks
1.1 Terminology and equipment
1.2 Internet organization
1.3 Layer models
2. Application Layer
2.1 Names and addressing schemes
2.2 DNS and HTTP protocols
3. Transport Layer
3.1 UDP protocol
3.2 Error Control, Flow and Congestion
3.2 TCP protocol and performance problems
4. Network Layer
4.1 IP Protocol
4.2 Addressing in IP networks (CIDR)
4.3 ARP, DHCP, ACLs and NAT
4. Static and dynamic routing (OSPF, BGP)
5. Data and physical link Layer
5.1 Introduction to modulation, bandwidth and communication media
5.2 IEEE 802 architecture
5.3 Ethernet (Switching, Link Aggregation, Spanning Trees, VLANs)
5.4 Local network topologies (data center and campus networks)

Teaching methodology and assessment:

The teaching/learning methodology is structured to promote a balanced combination of theory and practice, integrating different approaches to stimulate the active involvement of students and the application of acquired knowledge in real scenarios. The proposed methodology involves three main components: theoretical-practical classes, practical laboratory classes and group projects on the design and configuration of network topologies and protocols.

Assessment is structured on a continuous basis and covers different components, with the aim of guaranteeing a comprehensive understanding of the concepts and techniques covered throughout the course. This curricular unit requires a positive assessment in the Project to guarantee success in the course. Assessment also includes two tests during the academic semester and a compulsory comprehensive final exam.
Assessment Elements: 2 Tests, Project, Final Exam.

Bibliography:
[1] Computer Networking: A Top-Down Approach" de James Kurose e Keith Rose, 17h ed., Pearson 2016
[2] "Computer Networks, Global Edition" de Andrew S. Tanenbaum e David Wetherall, 6th ed, Pearson 2021
[3] "CCNA 200-301 Official Cert Guide Library Vol 1 & 2" de Wendem Odom et al., Cisco Press, 2024

Integrated Project Laboratory (3rd year, 7 ECTS) prerequisites

At the end of this curricular unit, the student should be able to:
1. Demonstrate knowledge and understanding of:
   - The creation of a project plan in project management.
   - Planning and controlling the project phases and the tasks assigned to each team member.
   - Identifying project risks and developing contingency plans.
   - Defining criteria and ensuring quality control.
2. Apply the acquired knowledge and understanding to:
   - Gain in-depth training in the area of Project Management, in accordance with the PMI (Project Management Institute) standards set out in the PMBOK (Project Management Body of Knowledge) Guide.
   - Develop skills appropriate to project management activities.
   - Prepare for professional activity through the acquisition of advanced knowledge in Project Management.

Syllabus:
1. Introduction.
1.1 Projects and Project management.
1.2 Why Projects Fail.
2. The Science of Project Management.
2.1 Organizational Influences and Project Management.
2.2 Standardization of Project Management.
2.3 The Project Life Cycle.
2.4 PMI Project Management.
2.5 The PMBOK Guide and Project Management Processes.
2.6 PMBOK and soft skills in a Project environment.
2.7 The creation and development of a Project team.
2.8 The challenges of communication in a Project environment.
2.9 Team decision-making techniques.
2.10 How to build trusting relationships with a Project's stakeholders.
3. Project Management.
3.1 Start the Project.
3.2 Organize and plan the Project.
3.3 Allocation of teams/resources, cost and risk management.
3.4 Execution, control and closure of the project.

3.5 Planning and Controlling Projects with Microsoft Office
3.6 Practical Case - Using MS Project as a tool to support Project planning and control.
4. Project Evaluation.
4.1 Project Monitoring.

Teaching methodology and assessment:
All classes will be exclusively practical for the development and management of the Project.
Projects will be prepared in groups, with a maximum of 3 people.
All Projects will be developed in the classroom, and can result from one of two situations:

1. are presented and proposed by the different professors supervising the subject or,
2. alternatively, if there are not enough proposals for the number of students, or when they do not accept any of the proposals presented, students will have to make a self-proposal.

At the end of the Project there will be an oral presentation/defense of each Project.
The final classification of the subject will be calculated as follows:
Project Classification = 50% of the classification given by the professor of the Integrated Project Laboratory subject + 50% of the classification given by the professor supervising the Project area.

Bibliography:
[1] Lewis, C., Lewis, CM. Chatfield, C., & Johnson, T. (2019). Microsoft Project 2019 Step by Step. Microsoft Press Store by Pearson.
[2] Biafore, B. (2013). Microsoft Project 2013: The Missing Manual. O'Reilly Media, Inc.
[3] Küster, J., Huber, E., Lippmann, R., Schmid, A., Schneider, E., Witschi, U., and Wüst, R. (2015). Project Management Manual. Springer Heidelberg New York Dordrecht London, doi: 10.1007/978-3-662-45373-5
It is also worth highlighting the need for development tools - programming IDEs for the specific programming languages to be used in the development of the Project. In this context, it may be necessary to search for appropriate bibliography adjusted to each of the realities that students will have to develop.

Multimedia II (3rd year, 6 ECTS) prerequisites

At the end of this curricular unit, the student should be able to:

- Describe modes and categories of compression techniques.
- Explain and analyse entropy encoding algorithms.
- Understand psychoacoustics principles and explain the functionality of source encoding methods for audio compression.
- Understand the Rate-Distortion theory and the functionality of source encoding methods based on image transforms/wavelets for image compression.
- Identify the main components of JPEG and MPEG standards.
- Propose, project and program the studied methods and algorithms as codecs.

Syllabus:
1. Compression fundamentals
1.1 Compression modes
1.2 Classification of compression techniques. Entropy and Source encoding
1.3 Topics for evaluating a compression system
2. Lossles compression methods and standards
2.1 Notion of Entropy
2.2 Generic model for variable length and fixed compression
2.3 Shannon-Fano and Huffman methods

2.4 Arithmetic coding
2.5 Lossles compression standards: JBIG, JPEG, JPEG-LS and PNG
3. Audio compression standards
3.1 Psychoacoustics and digitization of áudio
3.2 Standards for speech compression
3.3 Standards for Hi-Fi audio compression: MPEG-Audio, Dolby AC-3
4. Image compression fundamentals
4.1 Rate-distortion theory
4.2 Scalar quantizers
4.3 Basic coding schemes for lossy compression and DCT based compression
4.4 The JPEG standard: modes and architectures
4.4.1 Color processing
4.4.2 Quantization and entropy encoding
5. Project of a multimedia codec
5.1 Planning and design
5.2 Development and programming

Teaching methodology and assessment:
The methodology of teaching and learning is expository, interrogative and demonstrative. Drawing on problem solving and study geared to allow the interpretation of fundamental compression principles. Problems referring to audio, video and image compression are proposed, both as classroom work and individual study work.
Assessment: 2 tests (50% of final grade), script for a scientific video/tutorial about the proposed codec (10% of final grade), development of a scientific video/tutorial explaining the code for a codec (30% of final grade), presentation of the project (video and paper) (5% of final grade), student performance (5% of final grade)

Bibliography:
[1] Ribeiro, Nuno, Torres, José, Tecnologias de Compressão Multimédia, FCA – Editora de Informática, 2009.
[2] Sayood, K., Introduction to Data Compression, 5th Edition, Morgan Kaufman (Elsevier), 2017.
[3] McAnlis, C. & Haecky, A. Understanding Compression: Data Compression for Modern Developers, O'Reilly Media, 2016.
[4] Li, Ze-Nian, Drew, Mark S., Liu, Jiangchuan, Fundamentals of Multimedia (Texts in Computer Science), Second Edition, Springer, 2014.
[5] Salomon, D., Motta, G. Handbook of Data Compression. 5th Ed., Springer-Verlag, 2010.
[6] Ribeiro, Nuno M., Multimédia e Tecnologias Interativas, 5ª Edição Aumentada, FCA – Editora de Informática, 2012.

Distributed Systems (3rd year, 6 ECTS) prerequisites

At the end of this curricular unit, the student should be able to:

- acquire solid knowledge about the most common architectures, management models and protocols used in distributed systems.
- be able to define and characterize different types of distributed systems, namely the objectives, middleware services involved and architectural variants of the client-server base model.
- be able to characterize the most used techniques and patterns of communication in distributed systems, e.g., remote method invocation, asynchronous communication (message-based) and synchronous communication (streaming).

- be able to apply the concepts and implementation mechanisms of distributed systems by using existing technologies (e.g., invoking methods on remote objects (RMI) and Web Services).

Syllabus:
1. Introduction to Distributed Systems
2. Communication in Distributed Systems
3. Processing, Location & Synchronization of Resources
4. Replication of Resources and Fault Tolerance
5. Tools and Remote Services

Teaching methodology and assessment:
This course is organized into theoretic-practical (TP) and practical classes (PR).
The teaching methodology used within TP classes is based on the oral presentation of the contents as well as the discussion of the main issues about the covered topics. In parallel, the context of laboratory classes provide students the opportunity to contact with software tools (cf. RMI, Web Services) usually used for developing distributed systems. The assessment is continuous, contemplating with equal weight both the TP and PR components. The TP component is assessed by an exam at the end of the semester. The PR component is assessed through the planning and implementation of a project for a distributed system/service whose requirements are proposed during the semester.

Bibliography:
[1] Coulouris G., Dollimore J., Kindberg T., Blair, G. S., Distributed Systems Concepts and Design, 5th Ed., Addison-Wesley 2011.
[2] Tanenbaum, A., Steen, M., Distributed Systems, Principles and Paradigms, Prentice Hall 2002.
[3] Sun Developers Services, URL: http://developer.java.sun.com/developer/ (2009).
[4] Jian, Chen, Java RMI vs .Net Remoting, White Paper, 2003, http://students.cs.tamu.edu/jchen/cpsc689-608/comparison/l0c0607-framework_comp_lichen.pdf (2007).