Licenciatura: Engenharia Informática	**Área Científica:** 481 – Ciências Informáticas
Unidade curricular / Curricular Unit: Introdução à Ciência dos Computadores / Introduction to Computer Science	**ECTS:** 6
Duração: Semestral	**Horas de Contacto (Teórico Práticas):** 60 (48 TP + 12 OT)
Professor(a) responsável / Responsible academic staff	Doutor Helder Rodrigo Pinto
e-mail institucional	helder.pinto@my.istec.pt
Outros professores / Other academic staff	Dr. Sérgio Coelho

Objetivos de aprendizagem / Learning outcomes of the curricular unit (conhecimentos, aptidões e competências a desenvolver pelos estudantes)

No final os estudantes que a completarem deverão ser capazes de:

1. Escrever algoritmos e fluxogramas sobre problemas relacionados com o desenvolvimento de software.
2. Identificar requisitos funcionais e não funcionais para soluções e plataformas digitais.
3. Desenhar Diagramas de Casos de Uso, identificando os atores, funcionalidades e associações.
4. Gerir versões de código, num contexto de desenvolvimento de software colaborativo, recorrendo a ferramentas como Git.
5. Aplicar procedimentos de planeamento e gestão de projetos de software, recorrendo à ferramenta WBS (Work Breakdown Structure).
6. Trabalhar em equipa no desenvolvimento de um projeto de Engenharia Informática.
7. Resolver problemas de forma criativa, resiliente e autónoma.
8. Desenvolver a comunicação assertiva e o pensamento crítico.

(English)

Upon completion, students should be able to:

1. Write algorithms and flowcharts for problems related to software development.
2. Identify functional and non-functional requirements for digital solutions and platforms.
3. Design Use Case Diagrams, identifying actors, functionalities, and associations.
4. Manage code versions in a collaborative software development context using tools such as Git.
5. Apply software project planning and management procedures using the WBS (Work Breakdown Structure) tool.
6. Work as a team on the development of a Computer Engineering project.
7. Solve problems creatively, resiliently and autonomously.
8. Develop assertive communication and critical thinking.

Conteúdos programáticos / Syllabus

1. Introdução às Linguagens de Programação
   - 1.1. Origens e evolução
   - 1.2. Características
   - 1.3. Plataformas de desenvolvimento
   - 1.4. Linguagens compiladas e interpretadas
   - 1.5. Linguagens tipadas e não tipadas
2. Algoritmos e Fluxogramas
   - 2.1. Variáveis
   - 2.2. Operações
   - 2.3. Estrutura de decisão
   - 2.4. Estrutura de repetição
   - 2.5. Traçagem

3. Engenharia de Requisitos
    3.1. Requisitos funcionais
    3.2. Requisitos não funcionais
    3.3. Diagrama de Casos de Uso
4. Gestão de Versões de Código
    4.1. Git
    4.2. Github
    4.3. Branches
5. Gestão de Projetos
    5.1. Conceitos
    5.2. Metodologias
    5.3. WBS
    5.4. Deadlines
    5.5. Milestones
    5.6. Gestão de Recursos
6. Projeto de Engenharia Informática
    6.1. Planeamento
    6.2. Implementação Colaborativa
    6.3. Documentação

(English)

1. Introduction to Programming Languages
    1.1. Origins and evolution
    1.2. Characteristics
    1.3. Development platforms
    1.4. Compiled and interpreted languages
    1.5. Typed and untyped languages
2. Algorithms and Flowcharts
    2.1. Variables
    2.2. Operations
    2.3. Decision Structure
    2.4. Repetition Structure
    2.5. Tracing
3. Requirements Engineering
    3.1. Functional Requirements
    3.2. Non-functional Requirements
    3.3. Use Case Diagram
4. Code Version Management
    4.1. Git
    4.2. Github
    4.3. Branches
5. Project Management
    5.1. Concepts
    5.2. Methodologies
    5.3. WBS
    5.4. Deadlines
    5.5. Milestones
    5.6. Resource Management
6. Computer Engineering Project
    6.1. Planning
    6.2. Collaborative Implementation
    6.3. Documentation

Demonstração da coerência dos conteúdos programáticos com os objetivos da unidade curricular / Demonstration of the syllabus coherence with the curricular unit's objectives

A Unidade Curricular (UC) de Introdução à Ciência dos Computadores (ICCOM), do primeiro semestre do primeiro ano da Licenciatura em Engenharia Informática (LEI) do ISTEC Porto, adotará um método de ensino centrado no desenvolvimento de um projeto único interdisciplinar. Este método visa promover a integração e aplicação dos conhecimentos e competências adquiridos em várias UCs lecionadas neste semestre, sendo as mais relevantes neste contexto: Tecnologias de Internet I (TINT1); Programação I (PROG1); Matemática 1 (MATE1); e Arquitetura e Funcionamento dos Computadores (ARFUC).
Os objetivos programáticos, como escrita de algoritmos, identificação de requisitos, desenho de diagramas UML, gestão de versões (Git), planeamento (WBS), trabalho em equipa, resolução criativa de problemas e comunicação, permitem a aplicação prática, a colaboração e o desenvolvimento de competências técnicas e sociais essenciais para a Engenharia Informática, preparando os estudantes para desafios reais e consolidando a sua formação.

(English)

The Introduction to Computer Science (ICCOM) course unit, taught in the first semester of the first year of the Computer Engineering Degree (LEI) at ISTEC Porto, will adopt a teaching method focused on the development of a single interdisciplinary project. This method aims to promote the integration and application of the knowledge and skills acquired in various CUs taught this semester, the most relevant of which are: Internet Technologies I (TINT1); Programming I (PROG1); Mathematics 1 (MATE1); and Computer Architecture and Operation (ARFUC).
The programme objectives, such as algorithm writing, requirements identification, UML diagram design, version management (Git), planning (WBS), teamwork, creative problem solving and communication, enable the practical application, collaboration and development of technical and social skills essential for Computer Engineering, preparing students for real challenges and consolidating their training.

Metodologias de ensino e avaliação / Teaching methodologies including evaluation

As aulas desta UC assumem um carácter teórico-prático e são lecionadas num contexto baseado em casos práticos e projetos, complementadas com demonstração da aplicação prática com recurso a exercícios, bem como orientação e mentoria em trabalhos.

(English)

The classes in this CU are theoretical and practical in nature and are taught in a context based on practical cases and projects, complemented by demonstrations of practical application using exercises, as well as guidance and mentoring in assignments.

Demonstração da coerência das metodologias de ensino com os objetivos da unidade curricular / Demonstration of the coherence between the teaching methodologies and the learning outcomes

As metodologias de ensino da UC, assentes num carácter teórico-prático e fortemente alicerçadas em casos práticos e projetos, estão intrinsecamente alinhadas com os objetivos de aprendizagem, promovendo uma aplicação direta e consolidada dos conhecimentos.
A abordagem teórico-prática e baseada em projetos permite que os estudantes desenvolvam competências essenciais como a escrita de algoritmos, o desenho de diagramas UML (casos de uso) e a aplicação de engenharia de *software* (identificação de requisitos, planeamento com WBS). A demonstração prática com exercícios reforça a aquisição destas habilidades técnicas, assegurando que os estudantes as dominam ativamente.
A orientação e mentoria contínuas nos trabalhos são vitais para o desenvolvimento do trabalho em equipa, a gestão de versões com Git e a resolução criativa e autónoma de problemas inerentes a um projeto interdisciplinar. Esta supervisão facilita ainda o aperfeiçoamento da comunicação assertiva e do pensamento crítico, transformando a teoria em prática e capacitando os estudantes com as competências globais exigidas no contexto da Engenharia Informática.

(English)

UC's teaching methodologies, based on a theoretical-practical approach and strongly grounded in practical cases and projects, are intrinsically aligned with learning objectives, promoting a direct and consolidated application of knowledge.

The theoretical-practical and project-based approach allows students to develop essential skills such as writing algorithms, designing UML diagrams (use cases) and applying software engineering (identifying requirements, planning with WBS). Practical demonstrations with exercises reinforce the acquisition of these technical skills, ensuring that students actively master them.
Continuous guidance and mentoring in the work are vital for the development of teamwork, version management with Git, and the creative and autonomous resolution of problems inherent in an interdisciplinary project. This supervision also facilitates the improvement of assertive communication and critical thinking, transforming theory into practice and equipping students with the global skills required in the context of Computer Engineering.

Bibliografia / Bibliography

Fundamental

Documentação da UC
Portela, F., & Pereira, T. (2023). *Introdução à Algoritmia e Programação em Python*. FCA
Nunes, M., & O'Neill, H. (2004). Fundamentos de UML. FCA
Miguel, A. (2018). Gestão de Projetos de Software. FCA

Complementar / Complementary

Carriço, A. J. et al. (2008). Arquitetura e Funcionamento dos Computadores. Edições Chambel Lda.