Ficha de Unidade Curricular A3ES
Computação Distribuída
Mestrado em Engenharia Informática e de Computadores
2025-26

1. Designação da unidade curricular

[4117] Computação Distribuída / Distributed Computing

2. Sigla da área científica em que se insere	IC
3. Duração	Unidade Curricular Semestral
4. Horas de trabalho	162h 00m
5. Horas de contacto	Total: 60h 00m das quais T: 20h 00m | TP: 10h 00m | P: 30h 00m
6. % Horas de contacto a distância	Sem horas de contacto à distância
7. ECTS	6
8. Docente responsável e respetiva carga letiva na Unidade Curricular	[710] Luís Manuel da Costa Assunção | Horas Previstas: 60 horas
9. Outros docentes e respetivas cargas letivas na unidade curricular	[1551] José Manuel de Campos Lages Garcia Simão | Horas Previstas: 60 horas

10. Objetivos de aprendizagem e a sua compatibilidade com o método de ensino (conhecimentos, aptidões e competências a desenvolver pelos estudantes).

1. Descrever e discutir as vantagens, os problemas e os desafios que se colocam no desenvolvimento de aplicações usando o paradigma
da computação distribuída;
2. Conhecer os padrões de arquitetura, de interação e comunicação entre as partes das aplicações distribuídas;
3. Conhecer e aplicar algoritmos relacionados com Tempo (relógios lógicos), ordenação e coordenação de eventos e consensos em
computação distribuída;
4. Desenvolver aplicações distribuídas, com recurso a middleware e API para acesso a componentes e serviços remotos, usando modelos
de chamadas remotas;
5. Saber utilizar o alojamento de serviços em infraestruturas distribuídas, locais e em Clouds públicas com recurso a Máquinas Virtuais e
Contentores.

10. Intended Learning objectives and their compatibility with the teaching method (knowledge, skills and competences by the students).

1. Know how to describe and discuss the advantages, problems and challenges that arise in the development of applications using the
distributed computing paradigm;
2. Know the patterns of architecture, interaction, and communication between the parts of distributed applications;
3. Know and apply algorithms related to Time (logic clocks), ordering and coordination of events and consensus in distributed computing;
4. Develop distributed applications using middleware and API to access to remote components and services, doing remote invocations;
5. Know how to deploy services in distributed local infrastructures and in public Clouds using Virtual Machines and Containers.

11. Conteúdos programáticos

1. Caracterização, Potencialidades e Desafios: Comunicação; Concorrência; Resiliência; Escalabilidade; Elasticidade; Replicação;
Teorema CAP; 8 Falácias da Computação Distribuída;
2. Execução de componentes distribuídas: Máquinas físicas e virtuais; Containers; infraestruturas Cloud;
3. Middleware de comunicação e interação: Sockets; Objetos distribuídos (Java RMI); Remote Procedure Call (gRPC); Serviços REST e
Microservices; Callbacks, Comunicação assíncrona em streaming; Serviços Stateless/Stateful; Message Queuing; Publish/Subscribe e
event-driven (RabbitMQ);
4. Tempo, ordenação de eventos, coordenação e consensos. Relógios lógicos e vetoriais na ordenação causal de mensagens. Exclusão
mútua e consensos baseados em eleições distribuídas. Comunicação por grupos e multicast com eventos de membership (Spread Toolkit);
5. Concretização de aplicações em infraestruturas locais e infraestruturas Cloud. Experimentação com máquinas virtuais e Containers
(Google Cloud Platform e Docker).

11. Syllabus

1. Characterization, Potential and Challenges: Communication; Concurrency; Resilience; Scalability; Elasticity; Replication; CAP Theorem;
8 Fallacies of Distributed Computing;
2. Execution of the distributed components: Physical and Virtual machines; Containers; Cloud Infrastructures;
3. Middleware for communication and interaction: Sockets; Distributed Objects (Java RMI); Remote Procedure Call (gRPC), REST and
Microservices; Callbacks, Asynchronous communication and Streaming; Stateless/Stateful Components; Message Queuing;
Publish/Subscribe and event-driven (RabbitMQ);
4. Time, event ordering, coordination, and consensus. Logical and vector clocks for causal ordering of messages. Mutual exclusion and
consensus based on distributed elections. Group communication and multicast with membership events (Spread Toolkit);
5. Implementation of applications on local and Cloud infrastructures. Practical laboratory with Virtual Machines and Containers (Google
Cloud Platform and Docker).

12. Demonstração da coerência dos conteúdos programáticos com os objetivos de aprendizagem da unidade curricular

Tendo como grande objetivo alicerçar unidades curriculares posteriores que necessitam dos fundamentos essenciais na área da
computação distribuída, os objetivos enunciados podem ser sumarizados na aquisição de competências fundamentais, nomeadamente as
características, as diferenças e os desafios da computação distribuída face à computação centralizada. Assim, os conteúdos
programáticos (1), (3) e (4) contribuem para os objetivos (1), (2), (3) e (4), pois ao serem apresentados exemplos concretos usando
tecnologias que implementam os diversos modelos de interação entre as partes das aplicações distribuídas, o aluno adquire competências
que lhe permitem desenvolver soluções concretas avaliadas através da realização de trabalhos práticos. Para o objetivo (4) e (5)
contribuem os conteúdos programáticos (2) e (5), onde o aluno tem de demonstrar a funcionalidade das soluções desenvolvidas nos
trabalhos práticos.

12. Evidence of the syllabus coherence with the curricular unit´s intended learning outcomes

Having as a great objective to create a base used in later curricular units that need the essential foundations in the area of distributed
computing, the stated objectives can be summarized in the acquisition of fundamental competences, namely the characteristics, the
differences and the challenges of the distributed computing face to the centralized computing. Thus, the programmatic contents (1), (3) and
(4) contribute to objectives (1), (2), (3) and (4), since concrete examples are presented using technologies that implement the various
models of interaction between the parts of the distributed application, where students acquire skills that allow them to develop concrete
solutions evaluated through the accomplishment of practical works. For the objective (4) and (5), contribute the programmatic contents (2)
and (5), where the student is challenged to evaluate and to demonstrate the functionality of the solutions developed in practical works.

13. Metodologias de ensino e de aprendizagem específicas da unidade curricular articuladas com o modelo pedagógico

A metodologia aposta na autonomia do estudante na procura de soluções para problemas e desafios concretos que se colocam no
desenvolvimento de sistemas de computação distribuída.
A atual disponibilidade de infraestruturas de computação fortemente assentes em tecnologias de virtualização, nomeadamente as
oferecidas para educação pelos grandes fornecedores de Cloud permitem que cada grupo de alunos tenha disponível um laboratório de
computação distribuída autónomo, disponível 24x7 horas e totalmente gerido pelos alunos.
Assim, a metodologia de ensino/aprendizagem é centrada no estudante, sendo determinada pelo seu empenho em implementar sistemas
de computação distribuída integrando os conceitos fundamentais e os padrões de arquiteturas de software apresentados pelo professor na
sala de aula, ou outros que o estudante, com sentido critico, considere serem adequados, nomeadamente recorrendo a plataformas de
inteligência artificial.

13. Teaching and learning methodologies specific to the curricular unit articulated with the pedagogical model

The methodology focuses on student autonomy in finding solutions to concrete problems and challenges that arise in the development of
distributed computing systems.
The current availability of computing infrastructures heavily based on virtualization technologies, particularly those offered for education by
major Cloud providers, allows each group of students to have an autonomous distributed computing laboratory, available 24x7 hours and
fully managed by the students.
Thus, the teaching/learning methodology is student-centered, determined by their commitment to implementing distributed computing
systems integrating the fundamental concepts and software architecture patterns presented by the professor in the classroom, or others
that the student considers appropriate with critical thinking, including the use of artificial intelligence platforms.

14. Avaliação

A avaliação é distribuída com exame final. Os resultados de aprendizagem (1), (2) e (3) são avaliados através de um exame de avaliação individual. Os resultados de aprendizagem (4) e (5) são avaliados com dois trabalhos de projeto com relatório de concretização e a demonstração/discussão em sala de aula.
Assim a avaliação é composta por:
1) Nota de Exame final (NE)>=9,5), pedagogicamente fundamental, com duração de 2 horas;
2) Trabalho de projeto 1 (NP1): Desenvolvimento de um sistema distribuído cliente/servidor
3) Trabalho de projeto 2 (NP2>=9,5), pedagogicamente fundamental, com o desenvolvimento de um sistema distribuído envolvendo os
middleware estudados.
Nota final (NF)=0,5xNE+0,2xNP1+0,3xNP2.

14. Assessment

Assessment is distributed with a final exam. The learning outcomes (1), (2) and (3) are assessed through a individual assessment final
examination. The learning outcomes (4) and (5) are evaluated in laboratory work and two work projects including a completion report and a
classroom demonstration and discussion of its functionality.
Thus the assessment is composed by:
1) Individual final Exam Grade (EG)>=9.5), pedagogically fundamental, lasting 2 hours;
2) Work Project 1 Grade (P1G): Development of a client/server distributed system
3) Work Project 2 Grade (P2G>=9.5), pedagogically fundamental, consisting of the development of a distributed system involving all studied
middlewares
Final Grade (FG)=0.5xEG+0.2xP1G+0.3xP2G.

15. Demonstração da coerência das metodologias de ensino com os objetivos de aprendizagem da unidade curricular

Através das aulas teóricas (T) são apresentados conceitos, modelos, padrões e arquiteturas das aplicações baseadas no paradigma da
computação distribuída. Em aulas teórico/práticas (TP) os alunos são motivados a discutir e experimentar exemplos de concretização dos
conceitos envolvidos. A aposta sistemática de demonstrar e concretizar exemplos de aplicação com plataformas tecnológicas existentes,
contribui coerentemente para os objetivos de aprendizagem consolidado pela avaliação de trabalhos práticos (P) laboratoriais realizados
autonomamente pelos alunos.

15. Evidence of the teaching methodologies coherence with the curricular unit´s intended learning outcomes

Through theoretical classes (T), concepts, models, patterns, and architectures of applications based on the distributed computing paradigm
are presented. In theoretical/practical classes (TP), students are motivated to discuss and experiment with examples that implement the
concepts involved. The systematic approach of demonstrating and implementing application examples with existing technological platforms,
coherently contributes to the learning objectives, consolidated by the evaluation of practical (P) laboratory work carried out autonomously by
the students.

16. Bibliografia de consulta/existência obrigatória

1-George Coulouris, Jean Dollimore, Tim Kindberg and Gordon Blair (2011), Distributed Systems: Concepts and Design, Fifth Edition,
Addison Wesley
2-Letha Hughes Etzkorn (2017), Introduction to Middleware: Web Services, Object Components, and Cloud Computing, CRC Press
3-Martin Kleppmann (2017), Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems,
1st Edition

17. Observações

Unidade Curricular Obrigatória
Unidade Curricular comum ao(s) curso(s) de MEIM

Data de aprovação em CTC: Data de aprovação em CTC:

Data de aprovação em CP: Data de aprovação em CP: