

## Objetivo

O presente trabalho prático 2 (TP2) tem como objetivo desenvolver as capacidades dos estudantes na integração de diferentes serviços através da escrita de APIs, importação de dados e integração entre bases de dados e outros serviços. Leia atentamente todo o enunciado e contacte os docentes da disciplina em caso de dúvidas.

- Cada **grupo de trabalho** deverá ser **composto por os mesmos alunos que realizaram o TP1**. Não será permitida a formação de novos grupos e apenas poderão realizar o trabalho os grupos que submeteram o TP1.
- A submissão dos novos trabalhos será realizada via Moodle em formulário a indicar para o efeito.

## Regras

- Para o desenvolvimento dos trabalhos propostos, poderá ser utilizada a linguagem de programação Javascript com a framework Node.js.
- Os pormenores de implementação que se devem à interpretação dos enunciados por parte dos grupos de alunos deverão ser descritos nos slides com detalhe e justificação das opções tomadas.
- A implementação de funcionalidades extra não presentes no enunciado será valorizada, desde que estas funcionalidades não modifiquem os requisitos obrigatórios e não reduzam a dificuldade do trabalho. As funcionalidades extra implementadas deverão ser documentadas nos slides a entregar com o trabalho.
- A apresentação de relatórios e/ou implementações **não originais e que constituam plágio**, conduzem à imediata atribuição de **nota zero** no trabalho de grupo e a **eventuais processos disciplinares**.

## Avaliação e entrega

- O trabalho prático 2 faz parte da avaliação da componente prática da disciplina de **Integração de Sistemas** (correspondendo a 30% da nota final).

- A nota é atribuída individualmente aos elementos do grupo segundo a apresentação, visualização e discussão dos elementos entregues e as impressões obtidas pelos docentes acerca do aluno durante o decorrer das aulas de acompanhamento.
- Para aprovação à disciplina, a nota da Componente Prática deverá ter a classificação mínima obrigatória de **10.0 valores**.
- As discussões orais deste trabalho serão realizadas durante as aulas práticas em dia indicado no calendário e no horário previamente definido para o efeito. Os grupos de trabalho devem reservar o dia e hora marcados, na sua agenda. Os trabalhadores-estudantes devem, ao abrigo do respetivo estatuto, solicitar a folha justificativa de exame para a empresa.
- O trabalho prático deverá ser submetido através do Moodle seguindo as instruções lá indicadas. A entrega deverá conter os seguintes elementos:
  - Código fonte em Javascript;
  - Scripts de bases de dados ou de testes;
  - Slides em Powerpoint (seguindo estrutura a indicar);
  - Vídeo com duração máxima de 5 minutos com instruções de utilização e demonstração de funcionalidades;

## Descrição do trabalho

1. Cada grupo deverá utilizar o mesmo dataset utilizado no Trabalho Prático 1 (TP1) e reutilizar a estrutura já definida no trabalho anterior.
2. Deverão ser selecionadas pelo menos 3 entidades presentes no dataset do TP1, sendo que pelo menos 2 destas entidades deverão ter uma relação de pertença, isto é, uma das entidades deverá ter uma relação 1-N ou N-N com a 2ª entidade. Exemplos: Equipa / Jogador (uma Equipa tem vários jogadores, um jogador está em apenas uma equipa); Filme / Ator (cada ator pode participar em vários Filmes e cada Filme tem vários atores).
  - a. Se o dataset não permitir individualizar 3 entidades, o grupo deverá discutir com os docentes a procura de alternativas.
3. Deverá ser definida uma estrutura em base de dados NoSQL (MongoDB - [www.mongodb.com](http://www.mongodb.com)) onde serão representadas essas entidades. A base de dados deverá ter as coleções necessárias para representar as entidades definidas.

- a. Além das entidades do sistema, deverá ser criada uma coleção que represente os utilizadores do sistema. Os utilizadores apenas têm e-mail e password. As passwords dos utilizadores serão gravadas nesta base de dados com encriptação SHA512.
  - b. Os utilizadores terão 3 níveis de permissões:
    - i. **View** – apenas pode visualizar dados das entidades
    - ii. **Edit** – pode visualizar e editar dados das entidades do sistema
    - iii. **Admin** – tem permissões de Edit e pode fazer CRUD de utilizadores
  - c. Deverão ser implementados modelos utilizando tecnologias de Object Modeling. Recomenda-se a utilização de Mongoose [<https://mongoosejs.com/>].
  - d. Os identificadores únicos das entidades deverão usar chaves em GUID [<https://www.npmjs.com/package/guid>] em vez de chaves numéricas. Se as entidades originalmente já tinham ids de qualquer tipo, estes deverão ser convertidos para o novo formato.
4. Deverá ser criada uma API que permita realizar CRUD de todas as entidades e de utilizadores. Deverão ser igualmente criados endpoints que permitam relacionar entidades.
- a. A nomenclatura dos *endpoints* da API deverá seguir as normas REST. Deverão ser utilizados os estados e verbos HTTP mais adequados para as respostas e perguntas à API. Exemplos:
    - i. Obter um filme  
`GET {HOSTNAME}/api/movies/cc29b080-be7c-4e7f-b80c-76d47aee400c`
    - ii. Obter os atores de um filme.  
`GET {HOSTNAME}/api/movies/a8248e6f-7f58-4734-9768-e4fd287ef97f/actors`
    - iii. Adicionar um ator a um filme  
`POST {HOSTNAME}/api/movies/a8248e6f-7f58-4734-9768-e4fd287ef97f/actors`
    - iv. Atualizar um ator de um filme  
`PUT {HOSTNAME}/api/movies/a8248e6f-7f58-4734-9768-e4fd287ef97f/actors/8bb06980-53ff-4c13-b3fb-4887b3a4e935`
  - b. As operações de PUT deverão ser idempotentes.
  - c. A autenticação na API deverá utilizar JSONWEBTOKEN [<https://www.npmjs.com/package/jsonwebtoken>].
  - d. Todas as operações deverão ser validadas através dos modelos e respeitar as restrições de integridade e relações (ex: não deverá ser possível apagar um Ator que já esteja incluído em algum filme).

5. Na API deverão ser criados endpoints que permitam a migração dos dados do dataset anterior para MongoDB
  - a. O serviço Node.js deverá conectar-se diretamente à base de dados PostgreSQL definida anteriormente e obter os dados necessários para enviar para o MongoDB
  - b. Deverá ser criado um Endpoint para iniciar o trabalho de migração. Uma vez iniciada uma migração, este Endpoint ficará indisponível até que a migração esteja completa.
    - i. A migração deverá inserir todos os dados de todos os ficheiros não apagados
  - c. A migração de dados deverá ser idempotente, isto é, apenas importará entidades que ainda não foram gravadas anteriormente.
  - d. Deverá ser implementado um Endpoint que permita verificar o estado atual da migração. Este Endpoint retornar uma percentagem que indica a estimativa de trabalho já completo.
  - e.
6. Deverão ser implementados com recurso a GraphQL endpoints que permitam replicar todas as *queries* realizadas no trabalho anterior, mas sobre a nova estrutura. Os grupos que tenham queries mais simples poderão discutir queries alternativas com os docentes.
7. Deverão ser realizados testes simples a todos os endpoints. Poderá ser utilizado o Postman ou uma abordagem via código com Mocha [<https://mochajs.org/>].
8. Melhorias no trabalho anterior serão valorizadas. Todas as melhorias deverão ser indicadas no powerpoint a disponibilizar com o trabalho.