Trabalho Prático

Meta 1 - 5 valores

Pretende-se que seja desenvolvido um sistema distribuído para gestão e realização de perguntas de escolha múltipla, com uma única resposta certa, em contexto de aula. O sistema deverá suportar dois perfis de utilizador (docente e estudante), cada um com funcionalidades específicas. O sistema a desenvolver deve obedecer aos requisitos funcionais e arquiteturais definidos nas próximas secções e é composto por três tipos de componentes:

- **Cliente** – aplicação através da qual os utilizadores interagem com o sistema;
- **Servidor** – responsável pela lógica de negócio e gestão dos dados;
- **Serviço de diretoria** – responsável por manter uma lista de servidores ativos.

A implementação deve ser realizada em **Java**, aplicando os conceitos estudados ao longo da unidade curricular. Para efeitos de gestão e persistência da informação, os servidores devem recorrer a uma base de dados local **SQLite**.

1 Requisitos Funcionais

Um utilizador com perfil de docente pode:

- Criar e editar um registo pessoal (nome, e-mail e *password*), devendo, para o efeito, introduzir um código único (igual para todos os docentes), conhecido pelos docentes e armazenado na base de dados do sistema de forma segura (esquema baseado em *hash code*);
- Autenticar-se no sistema, fornecendo o e-mail e *password* como credenciais;
- Criar perguntas de escolha múltipla, definindo o enunciado, o número de opções, as opções, a opção correta e o período de disponibilidade (data/hora de início e de fim);
- Quando uma pergunta é criada, o sistema gera automaticamente um código de acesso que permite, aos utilizadores com perfil de estudante, visualizar e responder à pergunta durante o respetivo período;
- Editar os dados de uma pergunta, desde que não ainda tenha qualquer resposta associada;
- Eliminar uma pergunta, desde que ainda não tenha qualquer resposta associada;
- Consultar a lista de perguntas (ativas, futuras ou já expiradas) definidas pelo próprio, aplicando filtros de pesquisa (por exemplo: período de realização ou estado);
- Para uma pergunta expirada definida pelo docente, listar as respostas submetidas dos estudantes: dia e hora de realização; enunciado da pergunta; opções e opção certa; e, para cada resposta submetida, o número, o nome e o e-mail do estudante, bem como

a sua resposta. Podem, igualmente, ser apresentados dados estatísticos considerados relevantes (por exemplo, percentagem de respostas certas);

- No caso de uma pergunta expirada, exportar os resultados para um ficheiro em formato CSV, com a seguinte informação: dia e hora de realização; enunciado da pergunta; opções e opção certa; e, para cada resposta submetida, o número, o nome e o e-mail do estudante, bem como a sua resposta (ver Figura 1);
- Encerrar a sessão (*logout*).

```
"dia";"hora inicial";"hora final";"enunciado da pergunta";"opção certa"
"06-10-2025";"10:10";"10:12";"Para enviar e receber dados via TCP em Java, recorre-se a objetos do tipo:";"a"

"opção";"texto da opção"
"a";"Socket"
"b";"ServerSocket"
"c";"DatagramSocket"

"número de estudante"; "nome"; "e-mail";"resposta"
"123456";"Johan Brochet";"jobro@gmail.com";"a"
"123457";"Jeanne Duval";"duvalj@gmail.com";"a"
"123458";"Rodolphe Pillon";"rodo.pillon@gmail.com";"b"
```

Figura 1 – Exemplo hipotético de ficheiro CSV com a informação sobre uma pergunta com período expirado

Um utilizador com perfil de estudante pode:

- Criar e editar um registo pessoal (número de estudante, nome, e-mail e *password*), não sendo permitida a existência de registos com o mesmo número de estudante ou e-mail;
- Autenticar-se no sistema, fornecendo o e-mail e *password* como credenciais;
- Introduzir o código associado a uma pergunta (comunicado pelo docente em contexto de aula). Se o código for válido e a pergunta estiver no período de disponibilidade, esta será apresentada e o estudante poderá, então, selecionar uma resposta e submetê-la;
- Consultar as perguntas a que já respondeu e com período de disponibilidade expirado, incluindo se a resposta está certa ou errada. Podem ser aplicados filtros de pesquisa (por exemplo, por data de realização);
- Encerrar a sessão (*logout*).

Aspetos adicionais a considerar:

- As funcionalidades, com exceção do registo e autenticação, só estão disponíveis para utilizadores autenticados;
- Não podem ser criados utilizadores com emails ou números de estudante repetidos;
- Apenas utilizadores com o perfil de docente podem criar e gerir perguntas;
- As aplicações cliente, servidor e serviço de diretoria devem suportar a execução simultânea de diversas operações (notificações assíncronas, cessos concorrentes, etc.). Por esta razão, devem ser aplicados os mecanismos de programação concorrente estudados, sempre que necessário;
- Existem aspetos relacionados com funcionalidades e características arquiteturais/protocolares que estão omissos neste enunciado. Os grupos têm liberdade para tomar decisões de implementação adequadas. Em caso de dúvida, devem contactar um dos docentes.

2 Requisitos Arquiteturais e Protocolares

A arquitetura geral do sistema pretendido é apresentada na Figura 2, sendo constituída por um ou mais **servidores**, um **serviço de diretoria**, **clientes** e **bases de dados SQLite**.

Os servidores e o serviço de diretoria correm no mesmo domínio de difusão e formam um **cluster** que oferece o serviço pretendido (requisitos funcionais explicitados na Secção 1).

Cada servidor acede, de forma exclusiva, a uma base de dados SQLite local com a informação necessária ao funcionamento do sistema. Existindo mais do que um servidor, apenas um deles atende os clientes num determinado momento, enquanto os outros funcionam como servidores de *backup*, mantendo as suas bases de dados locais sincronizadas/consistentes com a do servidor principal. Este último fica responsável por propagar aos restantes servidores as atualizações aos dados resultantes das interações com os utilizadores (criação de perguntas, submissão de respostas, etc.).

Os princípios de funcionamento e de interação das aplicações **servidor**, **serviço de diretoria** e **clientes** são descritos nas secções seguintes. Em todos os cenários expostos, a comunicação pode ser feita da forma que os grupos de trabalho entenderem ser mais apropriada (ex.: estratégias baseadas em cadeias de caracteres ASCII ou em objetos serializados).

2.1 Servidores

- Devem ser lançados indicando, na linha de comando:
  - o endereço IP e porto UDP do serviço de diretoria;
  - o caminho da diretoria de armazenamento da base de dados SQLite local (ficheiro);
  - o endereço IP da interface de rede local usada para comunicação *multicast*.

- Na fase de arranque, um servidor começa por enviar, via UDP, um **pedido de registo** ao serviço de diretoria e aguardar pela **confirmação**. O pedido de registo deve incluir dois portos TCP automáticos usados pelo servidor: um para aceitar pedidos de ligação de clientes e outro para permitir que outros servidores se conectem e obtenham uma cópia da sua base de dados SQLite. O *datagrama* de confirmação deve incluir o endereço IP e o porto de escuta TCP automático, usado para obtenção da base de dados, do servidor registado há mais tempo no serviço de diretoria. O servidor assim identificado, que pode ser aquele que recebe a confirmação, é considerado o servidor principal atual, ou seja, aquele que atende os clientes.

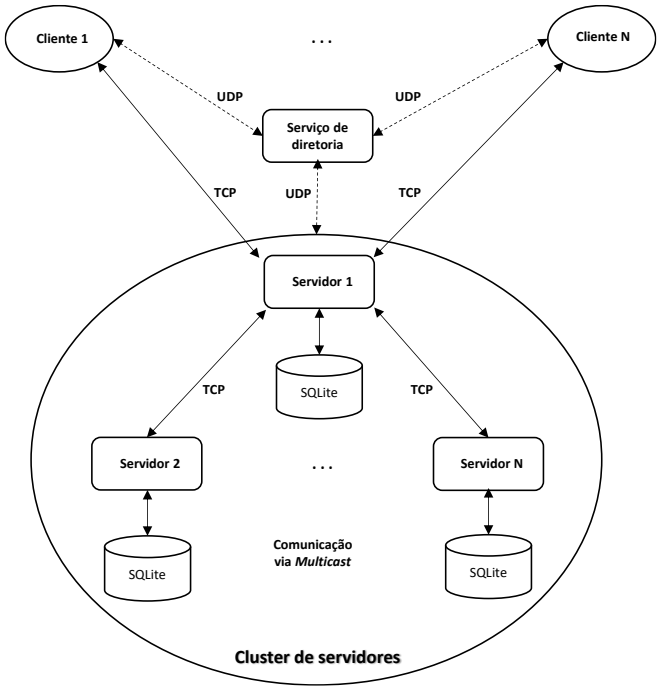

Figura 2 - Arquitetura geral do sistema

- Um servidor, que não receba qualquer resposta do serviço de diretoria durante a fase de arranque, termina.
- Na fase de arranque, se o servidor principal não possuir qualquer base de dados local criada (ficheiro com extensão ".db"), cria uma nova com número de versão 0 (estrutura criada, mas sem dados inseridos). Caso contrário, usa a base de dados com data de alteração mais recente existente na diretoria de armazenamento indicada através da linha de comando.
- Depois da resposta do serviço de diretoria, os servidores não principais começam por obter uma cópia integral da base de dados do servidor principal atual (ficheiro ".db"), via TCP, guardando-a localmente na diretoria indicada através da linha de comando. Se a operação não for bem-sucedida, termina a sua execução informando o serviço de diretoria.

- Durante uma operação de transferência da base de dados, que apenas ocorre durante a fase de arranque de um determinado servidor (não principal), deve ser garantido que não é feita qualquer alteração aos dados pelo servidor principal.
- Os servidores não devem apagar bases de dados SQLite locais existentes para evitar perdas *irreversíveis* de dados. Por esta razão, deve ser definido e implementado um mecanismo de nomeação dos ficheiros SQLite que permita criar bases de dados sem apagar as existentes, sempre que se justifique.
- Um servidor que não é o principal pode passar a sê-lo a qualquer momento, se o principal deixar de estar operacional. Por esta razão, todos os servidores aguardam continuamente por pedidos de ligação TCP de clientes no porto indicado.
- Cada base de dados SQLite tem um número de versão associado, incrementado sempre que é feita uma atualização (valor guardado numa tabela própria, conforme indicado no diagrama entidade-relacionamento da Figura 3).
- Depois da fase de arranque, um servidor envia, a cada **5 segundos**, uma mensagem de ***heartbeat*** para o **porto 3030** do endereço ***multicast* 230.30.30.30** e para o serviço de diretoria, que devolve um *datagrama* que inclui o endereço IP e o porto de escuta TCP (para obtenção da base de dados) do servidor registado há mais tempo no serviço de diretoria (ou seja, o principal atual).
- Os *heartbeats* incluem, no mínimo, a seguinte informação:
    - número de versão atual da base de dados local;
    - porto de escuta TCP automático no qual se aguarda por pedidos de conexão de clientes;
    - porto de escuta TCP automático no qual se aguarda por pedidos de ligação de outros servidores para obtenção de uma cópia integral da base de dados local.
- Um *heartbeat* também é emitido depois de ser feita qualquer alteração à base de dados local, com inclusão adicional da **query SQL** que foi executada (inserção, atualização ou eliminação). Exemplos: criação de uma pergunta, submissão de uma resposta, atualização de dados de um utilizador.
- Depois da fase de arranque, um servidor aguarda continuamente pela receção de *heartbeats* enviados para o porto 3030 do endereço de *multicast* 230.30.30.30. A receção de um *heartbeat* proveniente do próprio servidor ou de um servidor que não seja o principal é ignorada. Caso contrário:
    - **sem *query* SQL** incluída, mas com número de versão da base de dados diferente da versão da base de dados local, provoca o final da execução do servidor (indicador de perda de sincronização com a base de dados principal);
    - **com *query* SQL** incluída, mas com número de versão diferente do valor local acrescido de 1, leva igualmente ao final da execução do servidor (indicador de perda de sincronização com a base de dados principal);
    - **com *query* SQL** incluída e número de versão igual ao valor local acrescido de 1, leva à execução da *query* na base de dados local.

- Depois de o servidor principal proceder a qualquer atualização da sua base de dados local, deve notificar os seus clientes, conectados via TCP, para que, de um modo assíncrono, atualizem as suas vistas/informação apresentada aos respetivos utilizadores.
- Quando um servidor é encerrado de forma ordenada/intencional, deve enviar um *datagrama* ao serviço de diretoria, solicitando a anulação do seu registo.

2.2 Serviço de diretoria

- Gere uma lista de servidores ativos, ordenada pela ordem de registo no serviço de diretoria, e aguarda continuamente pela receção de *datagramas* enviados por clientes e servidores, num porto de escuta UDP passado na linha de comando.
- Quando recebe um *heartbeat* de um servidor que ainda não consta da lista ordenada de servidores ativos, ignora-o (servidor não registado).
- Em resposta a um pedido de um cliente, fornece os dados (endereço IP e porto TCP de escuta para aceitação de pedidos de conexão de clientes) do servidor ativo registado há mais tempo, ou seja, do primeiro da lista.
- Os dados relativos a um servidor são eliminados da lista se não for recebido qualquer *heartbeat* emitido por este ao fim de 17 segundos (valor fixo/*hardcoded*), mesmo que se seja o servidor principal atual, ou seja, o primeiro da lista/mais antigo.
- Para garantir a consistência/coerência do sistema distribuído a desenvolver, o serviço de diretoria é o primeiro que deve ser posto a correr no sistema. Se, por alguma razão, este deixar de funcionar, todos os servidores devem ser parados antes de se reiniciar o serviço de diretoria.

2.3 Clientes

- São lançados fornecendo o endereço e o porto de escuta UDP do serviço de diretoria através da linha de comando.
- Solicitam ao serviço de diretoria o endereço IP e porto de escuta TCP do servidor ao qual se devem ligar. Se esta operação não for bem-sucedida, a aplicação terminar.
- Depois, começam por solicitar o email e a *password* ao utilizador para efeitos de autenticação, ou o conjunto de dados necessários ao registo de um novo utilizador.
- Estabelecem uma ligação TCP com o servidor principal e enviam a informação de autenticação ou de registo.
- Quando a autenticação falha ou as credenciais não são enviadas no espaço de 30 segundos, o servidor encerra a ligação TCP com o cliente.
- Quando a conexão TCP com o servidor atual deixa de estar operacional, a aplicação cliente volta a solicitar ao serviço de diretoria os dados sobre o servidor principal atual. Se for diferente do anterior (que deixou de estar acessível), volta a ligar-se e a autenticar-se, sem envolver o utilizador e tentando passar esta situação de

recuperação de falha o mais despercebida possível. Se os dados corresponderem ao mesmo servidor, volta a tentar uma segunda vez 20 segundos depois. Caso a operação não seja bem-sucedida, a aplicação termina.

- A informação (mensagens) trocada entre os clientes e o servidor principal pode assumir a forma e estrutura que os grupos entenderem ser mais apropriada.
- As **vistas dos clientes** (menus ou interfaces gráficas) devem ser atualizadas de forma assíncrona, de acordo com notificações recebidas do servidor (ex.: quando uma nova pergunta passa a estar ativa/disponível ou quando o período de uma pergunta expira).
- O código fonte das aplicações cliente deve estar estruturado em dois componentes distintos:
  - **Vista** (interação com o utilizador);
  - **Comunicação** (envio e receção de mensagens).

3 Estrutura da Base de Dados

A Figura 3 apresenta uma sugestão para o modelo Entidade-Relacionamento (ER) das bases de dados SQLite acedidas pelos servidores, a partir do qual pode ser definido e implementado o modelo físico. Uma pergunta de escolha múltipla deve possuir pelo menos 2 opções.

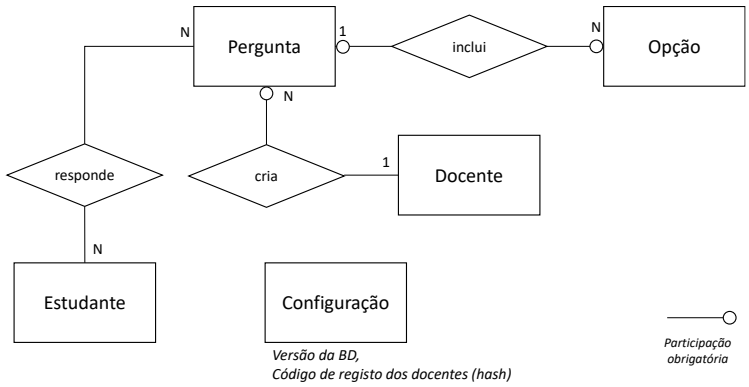

Figura 3 - Modelo Entidade-Relacionamento

4 Extras

A interface do utilizador da aplicação cliente descrita neste enunciado pode ser implementada em modo consola (i.e., texto). Os aspetos fundamentais considerados na avaliação base são se cumpre as funcionalidades pretendidas e se apresenta toda a informação necessária de forma adequada aos utilizadores. No entanto, os grupos que apresentem uma aplicação que cumpra minimamente os requisitos essenciais e que tenha uma interface do utilizador gráfica (GUI) funcional e completa, terão uma bonificação extra que poderá ir até aos 7.5% da nota atribuída. Ou seja, um trabalho avaliado em 80% que

tenha a totalidade deste extra passa a valer 86% (80% + 80% * 7.5%). Um trabalho avaliado em 100% não irá beneficiar da bonificação.

5 Considerações Gerais

Deve ter-se em consideração os seguintes aspetos:

- O trabalho deve ser realizado preferencialmente por grupos de **três** alunos, não podendo este valor ser ultrapassado;
- A constituição dos grupos deve ser registada através da plataforma Moodle;
- Cada grupo deverá utilizar o **GitHub** como sistema de controlo de versões ao longo do desenvolvimento do projeto. O repositório (privado) deve conter todo o código produzido e refletir a evolução do trabalho através do registo de *commits* e, caso se aplique, *merges* e *pull requests*, entre outras operações. Na defesa do projeto, cada grupo deverá apresentar o respetivo repositório, que servirá também para comprovar a participação individual dos elementos;
- Esta primeira meta do trabalho prático deverá ser entregue até ao dia **1 de dezembro de 2025**, **às 8h00**, através da plataforma InforEstudante, num ficheiro ZIP com a designação *PD-25-26-F1-TP-Gx.zip*, sendo *x* o número do grupo;
- Haverá uma **penalização de 10%** por cada hora de atraso na entrega;
- O ficheiro referido no ponto anterior deve incluir o código fonte (ficheiros ".java") e a documentação produzida, assim como os ficheiros auxiliares necessários à execução e teste das aplicações sem necessidade de recorrer a qualquer IDE (e.g., o *byte code* gerado e respetivas *batch files* e/ou ficheiros do tipo *jar* executáveis) – não devem ser entregues as pastas e respetivos conteúdos correspondentes ao projeto criado no IDE;
- As opções tomadas durante o projeto (e.g., aspetos não especificados no enunciado, variações devidamente justificadas ao nível das funcionalidades implementadas ou do modo de interação entre os diversos componentes, tratamento de anomalias, etc.), os aspetos relevantes do sistema desenvolvido (pormenores de implementação, diagramas temporais, estrutura/modelo físico da base de dados, etc.) e o manual de utilizador devem ser devidamente documentados de um modo sintético num documento do tipo PowerPoint;
- No documento referido na alínea anterior, é aconselhável a utilização de figuras e capturas de ecrã;
- Não é expectável que diferentes grupos apresentem soluções iguais. Caso este cenário se verifique, os grupos envolvidos terão de se justificar. **A deteção de situações de plágio leva a uma atribuição direta de 0 valores na nota do trabalho aos alunos de todos os grupos envolvidos**;
- Será valorizado o facto de o código das aplicações desenvolvidas ter sido estruturado de uma forma modular, com uma separação clara entre lógica de funcionamento, lógica de comunicação, lógica de acesso aos dados e interface do utilizador, podendo esta ser em modo texto ou gráfico conforme já mencionado.