Trabalho Prático

Meta 1 - 6 valores

Pretende-se que seja desenvolvido um sistema distribuído de reserva e compra de bilhetes para espetáculos que obedeça aos requisitos funcionais e arquiteturais definidos nas próximas secções. A implementação deve ser feita recorrendo à linguagem Java utilizando os conceitos estudados ao longo da unidade curricular. O sistema é composto por dois tipos de componentes: **cliente**, aplicação através da qual os utilizadores interagem com o sistema, e **servidor**, aplicação responsável por toda a lógica de negócio. Para efeitos de gestão e persistência de dados, um servidor recorre a uma base de dados SQLite. O sistema a desenvolver é composto por vários servidores que interagem entre si para garantir a disponibilidade do serviço oferecido através de mecanismos de redundância, replicação de dados e distribuição de carga. Também se pretende que as vistas nos clientes sejam automaticamente atualizadas quando uma alteração relevante ocorre no sistema.

1 Requisitos Funcionais

O sistema distribuído pretendido deve oferecer as seguintes funcionalidades:

- **Registo de novos utilizadores**, sendo estes caraterizados por um nome, um *username* e uma *password*;
- **Autenticação de utilizadores** já registados através do respetivo *username* e *password*;
- **Autenticação de um administrador** (*username*: `admin` e *password* inicial: `admin`);
- **Edição dos dados de registo** pelos utilizadores;
- **Consulta de reservas que aguardam confirmação de pagamento**;
- **Consulta de reservas pagas**;
- **Consulta e pesquisa de espetáculos** com base em diversos tipos de critérios/filtros (nome, localidade, género, data, etc.);
- **Seleção de um espetáculo** que irá decorrer pelo menos 24 horas depois da data atual;
- **Visualização dos lugares disponíveis e respetivos preços** no espetáculo selecionado (os preços são únicos, não existindo descontos previstos);
- **Seleção dos lugares pretendidos** (apenas na aplicação cliente, ou seja, sem notificação de qualquer servidor);
- Como o sistema permite acessos concorrentes, se a disponibilidade dos lugares se alterar durante o processo de escolha, a **informação visualizada nos clientes deve ser automaticamente atualizada**, o que pode implicar que lugares selecionados num cliente passem para o estado indisponível;
- **Submissão/validação de um pedido de reserva**, passando os lugares selecionados para o estado de indisponível no sistema. Se a operação falhar, o pedido é anulado pelo sistema e os respetivos lugares voltam a estar disponíveis;
- **Eliminação de uma reserva ainda não paga**;

- Depois de concluído e submetido com sucesso um pedido de reserva, o utilizador deve **proceder ao seu pagamento** dentre de um prazo limite (para efeitos de desenvolvimento, teste e defesa, pode definir-se um valor artificialmente baixo de, por exemplo, 10 segundos). Se a operação de pagamento falhar, o pedido é anulado pelo sistema e os respetivos lugares voltam a estar disponíveis;
- **Inserção de espetáculos** no sistema **pelo administrador**, através do carregamento de ficheiros de texto com a estrutura indicada na Figura 1. Qualquer problema identificado no ficheiro (campo desconhecido, identificadores de lugar duplicados, etc.) leva a que a operação seja ignorada de imediato, ficando sem efeito no sistema, e que a causa seja indicada ao administrador;

```
“Designação”;"Maria João & Carlos Bica Quarteto"
“Tipo”:”Música”
“Data”;"28";"10";”2022"
“Hora”;"21";”30"
“Duração”;”75”
“Local”;”Convento São Francisco – Sala D. Afonso Henriques”
“Localidade”;”Coimbra”
“País”;”Portugal”
“Classificação etária”;”6”

“Fila”;"Lugar:Preço”
“A”;”1:20”;”2:20”;”3:20”; “4:20”; “5:20”; “6:20”; “7:20”; “8:20” ”; “9:20”
“B”;”1:20”;”2:20”;”3:20”; “4:20”; “5:20”; “6:20”; “7:20”; “8:20”; “9:20”; “10:20”
“C”;”1:15”;”2:20”;”3:20”; “4:20”; “5:20”; “6:20”; “7:20”; “8:20”; “9:20”; “10:15”
“D”;”1:15”;”2:20”;”3:20”; “4:20”; “5:20”; “6:20”; “7:20”; “8:20”; “9:20”; “10:15”
“E”;”1:10”;”2:15”;”3:15”; “4:15”; “5:15”; “6:15”; “7:15”; “8:15”; “9:15”; “10:10”
“F”;”1:10”;”2:15”;”3:15”; “4:15”; “5:15”; “6:15”; “7:15”; “8:15”; “9:15”; “10:10”
“E”;”1:10”;”2:10”;”3:10”; “4:10”; “5:10”; “6:10”; “7:10”; “8:10”; “9:10”; “10:10”
“F”;”1:10”;”2:10”;”3:10”; “4:10”; “5:10”; “6:10”; “7:10”; “8:10”
“G”;”1:10”;”2:10”;”3:10”; “4:10”; “5:10”; “6:10”; “7:10”; “8:10”
“H”;”1:10”;”2:10”;”3:10”; “4:10”; “5:10”; “6:10”; “7:10”; “8:10”
```

Figura 1 – Exemplo de especificação de um espetáculo

- Depois de inserido no sistema, um espetáculo não fica imediatamente visível para os utilizadores comuns (apenas para o administrador). O administrador pode posteriormente alterar esta situação (torná-lo visível);
- **Eliminação de um espetáculo** por um administrador, desde que não haja qualquer reserva paga associada. No caso de uma eliminação de espetáculo, pedidos de reserva registados, mais ainda não pagos, são eliminados do sistema;
- ***Logout***.

Ao implementar as funcionalidades descritas, tenha em consideração os seguintes aspetos:

- As funcionalidades só estão disponíveis para utilizadores autenticados, o que significa que sem autenticação um utilizador não tem acesso ao sistema;
- Não podem ser criados utilizadores com nomes e *usernames* iguais;
- De um modo genérico, as aplicações cliente e servidor necessitam de suportar a execução simultânea de diversas operações (notificações assíncronas, etc.). Desta forma, deve recorrer-se aos mecanismos de programação concorrente estudados (e.g., *threads* e mecanismos de sincronização) sempre que se justificar a sua utilização;
- Existem aspetos relacionados com funcionalidades e características arquiteturais e protocolares que estão omissos neste enunciado. Os grupos de trabalho têm total liberdade para lidar com essas questões e implementarem soluções da forma que melhor entenderem. Em caso de dúvidas, devem contactar um dos docentes.

2 Requisitos Arquiteturais e Protocolares

A arquitetura geral do sistema pretendido é apresentada na Figura 2, sendo constituída por servidores, clientes e base de dados SQLite. Os servidores correm no mesmo domínio de difusão e formam um cluster que oferece o serviço pretendido (requisitos funcionais explicitados na secção 1) com características de tolerância a falhas e distribuição de carga. Cada servidor acede, de forma exclusiva, a uma base de dados SQLite local com a informação necessária ao funcionamento do sistema. Esta informação é replicada em todas as bases de dados. Isto implica que, quando um servidor atualiza a informação na sua base de dados, essa atualização deve propagar-se aos restantes servidores, sendo estes responsáveis por persistir a nova informação nas suas bases de dados para garantir a consistências das várias réplicas.

Os princípios de funcionamento e de interação das aplicações servidor e cliente 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. Por exemplo, podem ser utilizadas estratégias baseadas em cadeias de caracteres ASCII ou em objetos serializados, e na transferência de *queries* SQL, ficheiros ou outro tipo de informação.

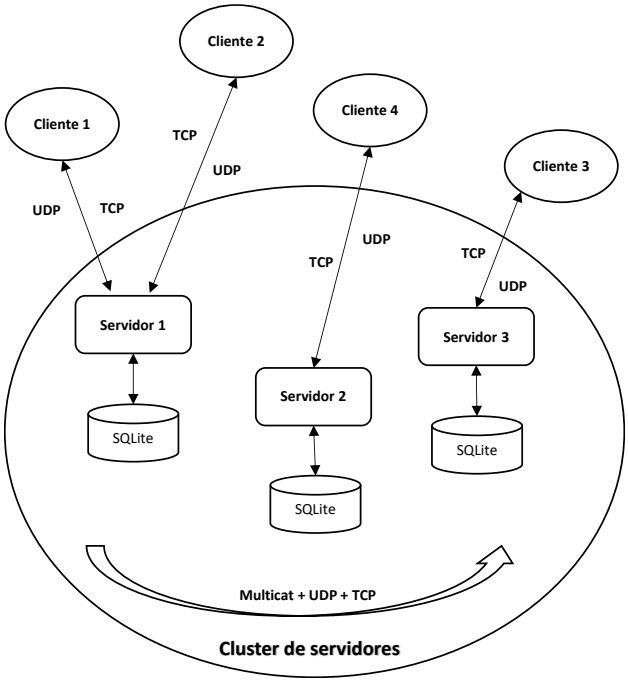

Figura 2 - Arquitetura geral do sistema

2.1 Servidores

- Todos os servidores que compõem o sistema situam-se no mesmo troço de rede (domínio de difusão), formando um cluster;
- Um servidor deve ser lançado indicando, na linha de comando, um porto de escuta UDP, onde irá aguardar por contactos de clientes, e o caminho da diretoria de armazenamento da sua base de dados SQLite;
- Um servidor aguarda continuamente pela receção de datagramas UDP enviados para o porto 4004 do endereço de *multicast* 239.39.39.39;
- Também aguarda continuamente por pedidos de ligação TCP num porto automático;
- Depois da fase de arranque, um servidor envia, a cada 10 segundos, uma mensagem de *heardbeat* para o porto 4004 do endereço de *multicast* 239.39.39.39;
- Os *heardbeats* incluem, no mínimo, a seguinte informação: porto de escuta TCP automático destinado a aceitar pedidos de ligação; estado de disponibilidade/indisponibilidade; número de versão da base dados local; e número de ligações TCP ativas (indicador de carga);
- Um *heardbeat* também é emitido quando ocorre qualquer mudança no servidor (número de clientes conectados, alteração na base de dados local, mudança de estado de disponibilidade, etc.);
- Com base na informação presente nos *heartbeats* recebidos, um servidor mantém uma lista atualizada dos servidores disponíveis no cluster, com as respetivas informações, e ordenada por ordem crescente em função da carga (número de ligações TCP ativas);

- Um servidor retira da sua lista de servidores disponíveis qualquer servidor cujo último *heartbeat* tenha sido recebido há 35 segundos ou mais, ou que indique estar indisponível num *heartbeat*;
- Uma base de dados local tem um número de versão associado que é incrementado sempre que é feita uma atualização;
- No arranque, um servidor entra numa fase de 30 segundos durante a qual apenas aguarda pela receção de *heartbeats.* Se não possuir qualquer base de dados local criada ou se verificar que existem outros servidores com bases de dados com números de versão superiores à sua, estabelece uma ligação TCP temporária com um dos servidores com a versão mais recente e com menos carga para iniciar um processo de criação de uma réplica local atualizada. Só depois é que inicia a difusão de *heartbeats* periódicos.
- Um servidor que, na fase de arranque, não possua qualquer base de dados local criada e não detete qualquer *heartbeat* durante 30 segundos cria uma base de dados nova com número de versão 1.
- Depois da fase de arranque, se um servidor detetar que a sua réplica da base de dados está desatualizada (número de versão inferior ao valor máximo comunicado pelos vizinhos nos seus *hearbeats*):
    - Interrompe qualquer operação em curso, ficando no estado de indisponível;
    - Fornece a lista atualizada e ordenada de servidores disponíveis aos clientes conectados;
    - Encerra as ligações TCP estabelecidas;
    - Envia um *heartbeat* através de *multicast* para informar de imediato os restantes servidores da sua alteração de estado;
    - Estabelece uma ligação TCP temporária com um dos servidores com a versão mais recente da base de dados e com menor carga para iniciar um processo de interação destinado a criar uma réplica local atualizada dos dados.
- Um servidor, quando recebe um *datagrama* UDP de um cliente a solicitar a sua ligação ao sistema, fornece uma lista de servidores **disponíveis** ordenada por ordem crescente pelo volume de carga (número total de ligações TCP ativas);
- Quando um servidor necessita de atualizar a sua base de dados local, é desencadeado um processo de troca de mensagens destinado a tentar manter a consistências das várias réplicas existentes no cluster de servidores:
    - Envia uma mensagem `Prepare` por *multicast* com informação suficiente para que seja efetuada a atualização pretendida, bem como o próximo número de versão da base de dados. A mensagem também inclui um porto automático UDP destinado a receber as confirmações;
    - Aguarda pela receção de mensagens de confirmação enviadas pelos restantes servidores disponíveis (via UDP *unicast* no porto indicado). Estas mensagens incluem o número indicado na mensagem `Prepare`. Um *timeout* de 1 segundo numa operação de receção indica o fim da espera por mensagens de confirmação;
    - Se não tiver recebido uma confirmação de todos os servidores disponíveis conhecidos, volta a enviar um segundo pedido. Se a situação se mantiver, a

operação de atualização deve ser cancelada em todos os servidores (incluindo no emissor do pedido). Para o efeito, envia, por *multicast*, a mensagem `Abort` no cluster de servidores, devendo esta incluir o número de versão inicialmente indicado na mensagem `Prepare`;
    - Depois de receber todas as confirmações esperadas, envia uma mensagem `Commit` por *multicast* com informação sobre o número de versão em causa;
    - Depois do envio/receção do `Commit`, todos os servidores, incluindo o emissor, efetivam a atualização das suas bases de dados locais SQLite, recorrendo à informação incluída na mensagem `Prepare` correspondente;
    - As mensagens `Commit` não necessitam de ser confirmadas pelos recetores.
- Em cada instante, apenas pode estar a decorrer uma única operação de atualização das bases de dados replicadas no sistema;
- Durante uma operação de atualização das bases de dados, não devem ser executadas operações de consulta;
- Depois de um servidor proceder à atualização da sua base de dados local, os seus clientes, conectados via TCP, devem ser notificados para que, de um modo assíncrono, procedam à atualização das suas vistas/informação apresentada aos utilizadores;
- Quando o nível de carga de um servidor ou a lista de servidores disponíveis se altera, o servidor fornece uma nova versão ordenada aos clientes conectados.

2.2 Clientes

- Um cliente é lançado fornecendo-lhe o endereço IP e o porto de escuta UDP de um dos servidores ativos;
- Começa por enviar, via UDP, um pedido de ligação ao servidor indicado;
- Como resultado, recebe uma lista de servidores (endereços IP e portos de escuta TCP) ordenada por ordem crescente em função da carga (ligações TCP ativas);
- Tenta estabelecer uma ligação TCP com o primeiro servidor da lista. Se falhar, tenta o segundo da lista e assim sucessivamente;
- Se, em qualquer momento, uma ligação TCP estabelecida entre um cliente e um servidor falhar por qualquer razão, o cliente volta a percorrer a lista de servidores, começando pelo primeiro, e tenta estabelecer uma nova ligação. As tentativas para contornar uma falha de ligação devem ser feitas de forma automática e da forma mais impercetível possível para os utilizadores das aplicações cliente;
- As vistas dos clientes devem ser atualizadas de forma assíncrona.

3 Estrutura da Base de Dados

A Figura 3 apresenta o modelo Entidade-Relacionamento (ER) das bases de dados acedidas por cada servidor, sendo a estrutura destas bases de dados igual em todos eles. Para que os alunos se possam focar nas componentes de programação distribuída, é fornecida juntamente com este enunciado uma base de dados SQLite que contem as tabelas, atributos

e relacionamentos necessários para suportar os requisitos funcionais do sistema aqui descrito. Nesse sentido, os alunos devem utilizar a estrutura da base de dados fornecida na implementação do trabalho prático. Para visualizarem essa estrutura, os alunos podem abrir a base de dados com um qualquer editor de SQLite (e.g., SQLiteStudio) ou podem consultar o modelo físico fornecido em anexo a este enunciado.

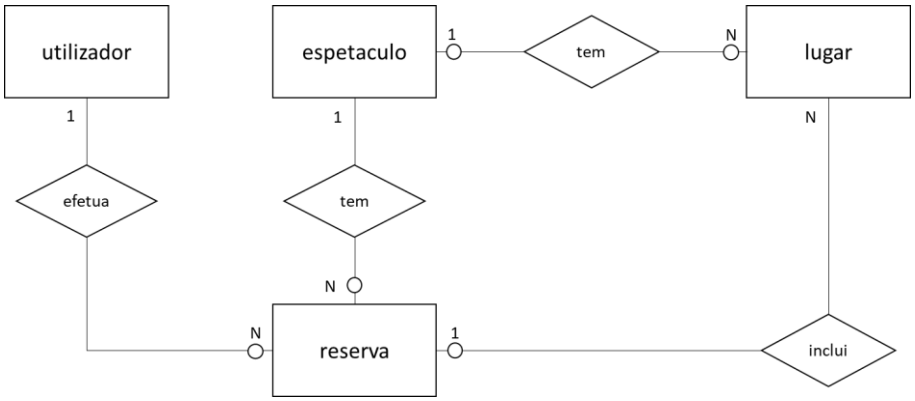

Figura 3 - Modelo ER das bases de dados

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%).

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 nem inferior a dois;
- A constituição dos grupos deverá ser registada em momento oportuno através da plataforma Moodle;
- Esta primeira meta do trabalho prático deverá ser entregue até ao dia **5 de dezembro de 2022**, **às 8h00**, através da plataforma InforEstudante, num ficheiro ZIP com a designação *PD-22-23-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);
- 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 das bases 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 e interface do utilizador, podendo esta ser em modo texto ou gráfico conforme já mencionado.