Trabalho Prático

Meta 1 - 6 valores

Pretende-se que seja desenvolvido um sistema distribuído de registo de presenças 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 três tipos de componentes: **cliente**, aplicação através da qual os utilizadores interagem com o sistema, **servidor principal**, aplicação responsável por toda a lógica de negócio, e **servidor de backup**, aplicação que interage com o servidor para manter uma cópia de segurança atualizada dos dados. Para efeitos de gestão e persistência de dados, os dois tipos de servidor recorrem a uma base de dados SQLite.

Em termos globais, o sistema pretendido deve possibilitar que qualquer indivíduo, com conhecimentos elementares de informática, o posso instalar e configurar para criar eventos e permitir que qualquer pessoa possa registar a sua presença durante o decorrer dos mesmos através de um código específico gerado e divulgado no momento.

1 Requisitos Funcionais

O sistema distribuído pretendido deve oferecer as seguintes funcionalidades aos **utilizadores da aplicação cliente**:

- **Registo de um novo utilizador**, sendo este caraterizado por um nome, um número de identificação (cartão de cidadão, número de identificação fiscal, número de estudante, etc.), um endereço de email e uma *password*. O endereço de email deve ser único e serve de *username*;
- **Autenticação de um utilizador** já registados através do respetivo endereço de email e *password*;
- **Edição dos dados de registo;**
- **Submissão do código associado a um evento para efeitos de registo da presença**. Se for feito fora do período de validade do código ou de realização do evento, a operação falha. O mesmo ocorre se um utilizador introduzir um código relativo a um determinado evento e já possuir presença registada noutro evento a decorrer naquele momento;
- **Consulta de presenças registadas**, podendo ser aplicados diversos tipos de critérios/filtros (período, nome do evento, etc.);
- **Obtenção de um ficheiro csv** com o resultado da consulta no ponto anterior (ver exemplo de base na Figura 1);
- ***Logout***.

```
“Nome”;“Número identificação”;"Email”
“João Silva”;”202011111”;”joao.silva@isec.pt”

“Designação”;“Local”;”Data”;“Hora início”
”Evento 2”;”DEIS/ISEC, Coimbra“;”10/10/2023”;”14:30”
”Evento 1”;”Convento de S. Francisco, Coimbra“;”9/08/2023”;”17:30”
```

Figura 1 – Consulta das presenças registadas por um determinado utilizador

O sistema distribuído pretendido deve oferecer as seguintes funcionalidades ao **administrador**:

- **Autenticação do administrador** (*username* e *password* pré-definidos na base de dados);
- **Criação de um evento,** sendo este caracterizado pelo nome, local, data de realização, hora de início e hora de fim**;**
- **Edição dos dados de um evento**. O período de realização apenas pode ser alterado se não possuir qualquer presença registada**;**
- **Eliminação de um evento**, desde que ainda não tenha qualquer presença registada;
- **Consulta dos eventos criados**, podendo ser aplicados diversos tipos de critérios/filtros (período, nome do evento, etc.);
- **Geração de um código para registo de presenças** em um evento que esteja a decorrer no momento, com indicação do tempo de validade em minutos. Podem ser gerados códigos sucessivos para o mesmo evento, prevalecendo o mais recente (os anteriores deixam de ser válidos);
- **Consulta das presenças registadas** num determinado evento;
- **Obtenção de um ficheiro csv** com a informação obtida no ponto anterior (ver exemplo de base na Figura 2);
- **Consulta dos eventos** em que um determinado utilizador, identificado através do seu email, tem presenças registadas;
- **Obtenção de um ficheiro csv** com a informação obtida no ponto anterior (ver exemplo de base na Figura 1);
- **Eliminação de presenças registadas** num determinado evento, sendo os utilizadores identificados através dos respetivos emails;
- **Inserção de presenças** num determinado evento, sendo os utilizadores identificados através dos respetivos emails;
- **Atualização assíncrona da informação visualizada** sempre que esta sofre alterações no servidor.
- ***Logout***.

```
“Designação”;" PD 2023/24 – TP1 – Aula 1"
“Local”:”DEIS/ISEC – L1.7, Coimbra”
“Data”;"12";"10";”2023"
“Hora início”;"14";”30"
“Hora fim”;”17”;”30”

“Nome”;“Número identificação”;"Email”
“João Silva”;”202011111”;”joao.silva@isec.pt”
“Joana Sila”;”202011122”;”joana.sila@isec.pt”
```

Figura 2 – Consulta das presenças registadas num determinado evento

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

- As funcionalidades, com exceção do registo e autenticação, só estão disponíveis para utilizadores autenticados;
- Não podem ser criados utilizadores com emails 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 3, sendo constituída por um servidor principal, eventuais servidores de backup, clientes e bases de dados SQLite (a principal no servidor principal e réplicas desta nos servidores de backup). Os dois tipos de servidor 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. Esta informação é atualizada pelo servidor principal, na sequência das interações com os utilizadores, e propagada às restantes bases de dados, que devem manter-se consistentes com a principal.

Os princípios de funcionamento e de interação das aplicações servidor principal, servidor de backup 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.

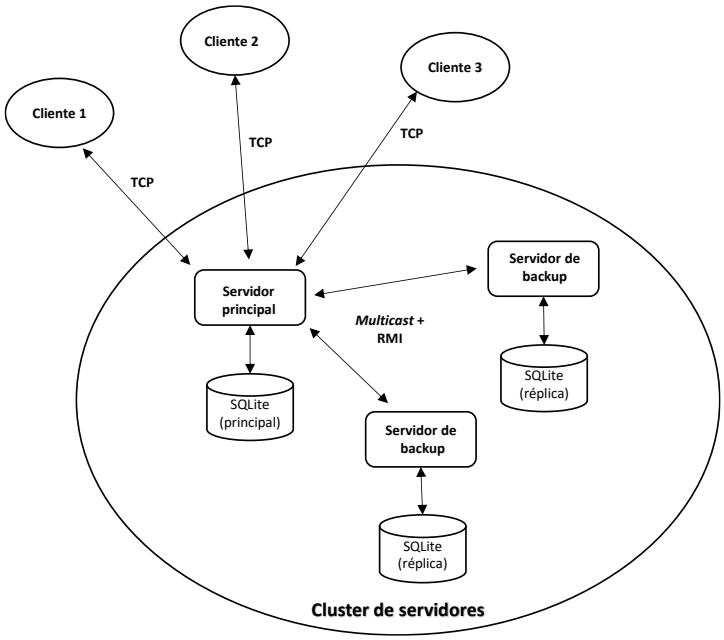

Figura 3 - Arquitetura geral do sistema

2.1 Servidor principal

- Deve ser lançado indicando, na linha de comando:
  - um porto de escuta TCP, onde irá aguardar pela conexão de clientes;
  - o caminho da diretoria de armazenamento da sua base de dados SQLite;
  - o nome com o qual deve registar um serviço RMI, destinado a interagir com servidores de backup, no *registry* local. Este serviço deve incluir, entre outras funcionalidades, a possibilidade dos servidores de backup se registarem para efeitos de atualização das suas bases de dados (réplicas da principal) através de um mecanismo de *callback*;
  - o porto de escuta no qual deve lançar o *registry* local;
- Aguarda continuamente por pedidos de ligação TCP de clientes no porto indicado;
- Depois da fase de arranque, um servidor envia, a cada 10 segundos, uma mensagem de *heardbeat* para o porto 4444 do endereço de *multicast* 230.44.44.44;
- Os *heartbeats* incluem, no mínimo, a seguinte informação: porto de escuta do *registry*, nome de registo do seu serviço RMI no *registry* local e número de versão da base dados local;
- Um *heardbeat* também é emitido depois de ser feita qualquer alteração à base de dados local e às bases de dados dos servidores de backup;
- Uma base de dados local tem um número de versão associado que é incrementado sempre que é feita uma atualização;
- Se, na fase de arranque, o servidor principal não possuir qualquer base de dados local criada, cria uma nova com número de versão 0 (estrutura criada, mas sem dados inseridos).

- Quando o servidor necessita de atualizar a sua base de dados local, também atualiza, através de RMI, as réplicas nos servidores de backup que estão registados no seu serviço RMI para efeitos de *callback*, desde que estas correspondam ao número de versão correto (ou seja, coincidente com a sua base de dados local). Servidores de backup com versões não coincidentes da base de dados são eliminados da lista de servidores de backup no servidor principal, deixando estes de ser notificados. Também ocorre o mesmo para os servidores de backup em que ocorre qualquer problema durante o processo de atualização das suas réplicas;
- Em cada instante, apenas pode estar a decorrer uma única operação de atualização das bases de dados no sistema;
- Depois de um servidor proceder à atualização da sua base de dados local e às réplicas nos servidores de backup, os seus clientes, conectados via TCP, devem ser notificados para que, de um modo assíncrono, procedam à atualização das suas vistas/informação apresentada aos respetivos utilizadores.

2.2 Servidores de backup

- Situam-se no mesmo troço de rede (domínio de difusão) do servidor principal, formando um *cluster*;
- São lançados indicando, na linha de comando, o caminho da diretoria para armazenamento da uma réplica da base de dados existente no servidor principal. Se a diretoria não estiver vazia, a aplicação deve terminar;
- Aguardam continuamente pela receção de *heartbeats* enviados pelo servidor principal para o porto 4444 do endereço de *multicast* 230.44.44.44. Se não detetarem qualquer *heartbeat* durante 30 segundos, terminam;
- Na fase de arranque:
    - começam por obter, através do serviço RMI do servidor principal, uma cópia integral da base de dados do servidor principal, guardando-a localmente num ficheiro com o nome indicado através da linha de comando. Durante uma operação de transferência integral de base de dados, deve ser garantido que não é feita qualquer alteração aos dados pelo servidor principal;
    - Depois da operação indicada no ponto anterior, registam no servidor principal, para efeitos de *callback*, o serviço RMI que implementam;
- Depois da fase de arranque, a receção de um *heartbeat* com número de versão da base de dados diferente do local provoca o final da execução do servidor de backup.

2.3 Clientes

- São lançados fornecendo o endereço e o porto de escuta TCP do servidor principal através da linha de comando;
- Começam por solicitar o *username* (email) e a password ao utilizador;
- Estabelecem uma ligação TCP com o servidor principal e enviam a informação de autenticação. Quando a autenticação falha ou as credenciais não são enviadas no espaço de 10 segundos, o servidor encerra a ligação TCP;

- A informação (mensagens) trocada entre os clientes e o servidor principal pode assumir a forma que os grupos de trabalho entenderem ser mais apropriada;
- As vistas dos clientes, que podem ser em modo texto ou gráfico, devem ser atualizadas de forma assíncrona.
- O código dos clientes deve estar estruturado em dois componentes distintos: vista (interação como o utilizador, menus, etc.) e comunicação (gestão da ligação TCP, envio e receção de mensagens, etc.).

3 Estrutura da Base de Dados

A Figura 4 apresenta uma sugestão para o modelo Entidade-Relacionamento (ER) das bases de dados SQLite acedidas pelos dois tipos de servidor, a partir do qual pode ser definido e implementado o modelo físico.

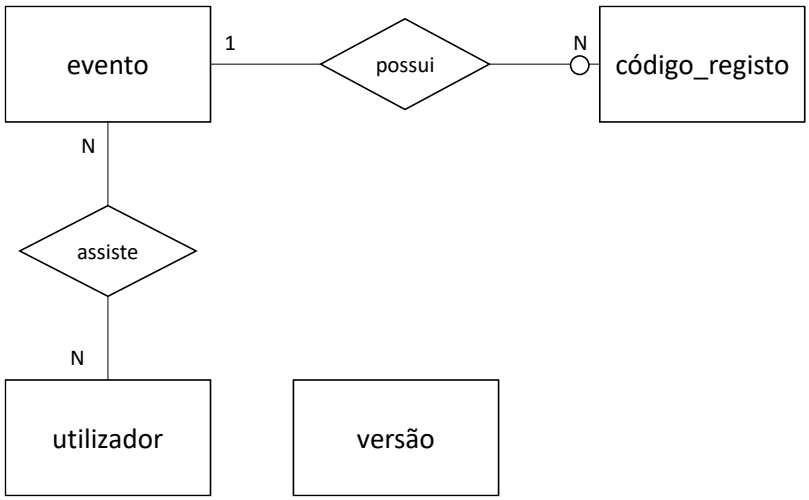

Figura 4 - Modelo ER das bases de dados

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;
- 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 **27 de novembro de 2023**, **às 8h00**, através da plataforma InforEstudante, num ficheiro ZIP com a designação *PD-23-24-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/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.