INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

Projecto de Programação com Objectos

Gestor de uma Rede Social

24 de Outubro de 2011

Esclarecimento de dúvidas:

- Consultar sempre o corpo docente atempadamente: presencialmente ou através do endereço **po11@l2f.inesc-id.pt**.
- Não utilizar fontes de informação não oficialmente associadas ao corpo docente (podem colocar em causa a aprovação à disciplina).
- Não são aceites justificações para violações destes conselhos: quaisquer consequências nefastas são da responsabilidade do aluno.

Requisitos para desenvolvimento, material de apoio e actualizações do enunciado (ver informação completa na secção **[Projecto]** no Fénix):

- O material de apoio é de uso obrigatório e não pode ser alterado.
- Verificar atempadamente (mínimo de 48 horas antes do final de cada prazo) os requisitos exigidos pelo processo de desenvolvimento.

Processo de avaliação (ver informação completa nas secções **[Projecto]** e **[Método de Avaliação]** no Fénix):

- Datas: **2011/10/27 12:00** (UML); **2011/11/15 12:00** (intercalar); **2011/12/02 12:00** (final); **2011/12/12–2011/12/16** (teste prático).
- Os diagramas UML são entregues exclusivamente em papel (impressos ou manuscritos) em local a anunciar. Diagramas ilegíveis serão sumariamente ignorados.
- Verificar atempadamente (mínimo de 48 horas antes do final de cada prazo) os requisitos exigidos pelo processo de avaliação, incluindo a capacidade de acesso ao repositório CVS.
- **Apenas se consideram para avaliação os projectos existentes no repositório CVS oficial.**
- Trabalhos não presentes no repositório no final do prazo têm classificação 0 (zero) (não são aceites outras formas de entrega). Não são admitidas justificações para atrasos em sincronizações do repositório. A indisponibilidade temporária do repositório, desde que inferior a 24 horas, não justifica atrasos na submissão de um trabalho.
- A avaliação do projecto pressupõe o compromisso de honra de que o trabalho correspondente foi realizado pelos alunos correspondentes ao grupo de avaliação. **Fraudes na execução do projecto terão como resultado a exclusão dos alunos implicados do processo de avaliação em 2011/2012**.

O objectivo do projecto é desenvolver os conceitos de suporte para uma rede social e aplicações que permitem geri-la e utilizá-la. A aplicação de gestão realiza operações de administração dos agentes que são os nós da rede, permite fazer consultas e guardar o estado persistentemente. A aplicação de navegação permite a interacção de agentes na rede e a publicação de conteúdos.

O projecto deve ser concebido de forma a possibilitar futuras extensões ou alterações de funcionalidade com um impacto mínimo no código pré-existente. Assim, deverá ser simples (i) gerir várias redes sociais; (ii) adicionar novos tipos de agente; (iii) adicionar novos tipos de publicação; (iv) adicionar novas políticas de acesso: e (v) adicionar novas formas de consulta.

1 Entidades, Propriedades, Funcionalidade

1.1 Entidades

Existem dois tipos de agentes (nós na rede): (i) agentes individuais, que podem estabelecer ligações com outros agentes individuais; e (ii) agentes organizacionais, que podem ser alvos de pedidos de ligações (mas não podem originá-las). Assim, existem dois tipos de ligações na rede: (i) relações mútuas (contactos) – apenas possíveis entre agentes individuais; e (ii) relações de interesse – tem como origem um agente individual e alvo uma organização.

Cada agente mantém informação sobre um conjunto de propriedades que o definem (§1.2.1) e gere uma colecção de publicações: notas textuais, URIs e imagens. Este elenco de entidades é apenas uma situação inicial (§1.2.2) e podem ser adionados novos tipos de entidades à colecção do agente.

Todos os agentes podem definir vários conjuntos de permissões, relativas a acções gerais (e.g., recepção de mensagens, estabelecimento de novas ligações), ou relativas a publicações – e.g., permissões para comentar/votar uma publicação (§1.2.2 e §1.4).

Os agentes podem ainda associar permissões às suas colecções ou apenas a elementos individuais. Se uma colecção tiver condições de acesso mais restritivas que as de um dos seus elementos, a permissão mais restritiva define as condições de acesso.

Todos os nós da rede podem trocar mensagens com nós com os quais partilhem ligações. Os agentes individuais podem limitar a recepção de mensagens através da definição de permissões de acesso. Estas permissões são definidas caso a caso (controlado o acesso de agentes individuais), ou através da definição de grupos de acesso (podem ser usados para agregar permissões de nós que partilham os mesmos direitos). Os controlos de acesso são positivos (caso normal), mas também podem ser negativos (para definir excepções): por exemplo, para permitir o acesso de omissão de um grupo e excluir apenas um dos seus elementos. Tal como nos casos positivos, a definição de excepções é rica e permite a definição de grupos com restrições (privilégios negativos).

Cada agente tem duas caixas de correio: uma para mensagens enviadas (caixa de saída: "outbox"), outra para mensagens recebidas (caixa de entrada: "inbox").

Todas as entidades podem escolher não impor qualquer tipo de restrição aos comentários aos elementos das suas colecções ou ao envio de mensagens para as quais são destinatários.

1.2 Propriedades

Todas as entidades da rede, excepto ligações (§1.2.4), possuem identificadores únicos (números inteiros). Sempre que for necessário produzir listas, a ordem é determinada pelos identificadores.

O identificador `0` (zero) é reservado e corresponde ao agente individual pré-definido de nome `root` (em minúsculas). Este identificador é utilizado para operações que exponham o funcionamento do sistema aos agentes (e.g., envio de mensagens comunicando problemas). Este agente tem sempre acesso a toda a informação do sistema.

Os identificadores têm um espaço único, i.e., se o identificador `12` (por exemplo) for usado por um agente, então não pode ser usado para identificar outro agente, outra mensagem, ou qualquer outra entidade.

1.2.1 Propriedades dos agentes

Além do identificador único, cada agente possui ainda um nome (cadeia de caracteres) e contactos: email (cadeia de caracteres); número de telefone (cadeia de caracteres numéricos, espaços, hífens, e o sinal +).

1.2.2 Propriedades das publicações

Além do identificador único, cada publicação contém a identificação do agente ao qual pertence e uma legenda.

Para os casos descritos acima (notas textuais, ligações WWW e imagens), as propriedades adicionais são, respectivamente: o conteúdo textual; a ligação WWW; e, para a imagem, o conteúdo gráfico que é armazenado (apenas por simplicidade) como uma cadeia de texto que representa a imagem (e.g., em lugar da imagem de uma bola, guardam-se os caracteres `[bola]`).

Todas as publicações possuem uma lista de comentários e uma pontuação (votos positivos e negativos): a lista de comentários e a votação estão limitadas por um conjunto de permissões definidas pelo agente. As publicações podem formar álbuns temáticos (que agrupam qualquer tipo de publicação, incluindo outros álbuns). Tal como as publicações individuais, os álbuns também podem ser alvo de comentários e votações. Apenas os agentes individuais podem comentar ou votar.

Uma vez criada, uma publicação não pode ser alterada (excepto através da adição de comentários e da contabilização de votações).

1.2.3 Propriedades das Mensagens

Além do identificador único, cada mensagem contém a identificação do emissor e dos receptores. Além dos campos identificativos, existe um campo para o assunto (cadeia de caracteres), um campo para o texto da mensagem (cadeia de caracteres) e uma lista de anexos (possivelmente vazia). Os anexos podem ser publicações ou outras mensagens.

1.2.4 Propriedades das ligações

As ligações são identificadas por dois identificadores de agentes, cada um correspondendo ao agente em cada extremo da ligação. A ligação pode estar activa (o receptor autorizou a ligação), inactiva (ainda não foi autorizada), ou bloqueada (a ligação foi autorizada, mas alguns tipos de interacção entre agentes podem não estar disponíveis).

As ligações desempenham um papel importante na propagação de mensagens e notificações entre os nós da rede, colaborando, entre outras actividades na verificação de algumas permissões.

1.3 Operações sobre agentes

Podem realizar-se as seguintes operações relativas a agentes: (i) visualizar; (ii) registar novos; e, (iii) activar ou desactivar.

Quando se regista um agente, ele fica activo e todas as condições de acesso são permissivas.

Além das operações de gestão, é ainda possível realizar outras operações de inspecção das propriedades e das colecções detidas pelos agentes (ver abaixo).

Os agentes não podem ser removidos, mas podem ser desactivados. Neste estado, toda a informação relativa ao agente mantém os acessos de leitura, mas todas as permissões de alteração são negadas, embora não alteradas. Assim, é possível manter um agente inalterado e reactivá-lo numa data posterior.

Os agentes podem alterar as suas propriedades, mas não o seu identificador único.

1.4 Operações sobre publicações

Podem realizar-se as seguintes operações relativas a publicações: (i) visualizar; (ii) registar novas; (iii) comentar e pontuar; e, (iv) proteger ou desproteger.

Todas as publicações estão desprotegidas quando são registadas, devendo ser protegidas caso a caso.

Quando se comenta ou vota uma publicação, além do registo apropriado à acção, é ainda registado o identificador do agente que executa a acção.

Quando se protege uma publicação, os agentes excluídos não podem aceder a nenhuma informação relativa à publicação indicada (incluindo pontuar ou comentar). A protecção de uma publicação protegida não tem qualquer efeito.

Quando se desprotege uma publicação, os agentes em causa podem aceder à informação relativa à publicação indicada (incluindo pontuar ou comentar). A remoção da protecção de uma publicação não protegida não tem qualquer efeito.

1.5 Operações sobre mensagens

Podem realizar-se as seguintes operações relativas a mensagens: (i) visualizar (recebidas e enviadas, ou uma mensagem individual); (ii) enviar (nova, resposta, reenvio); (iii) operar sobre uma mensagem (remoção, gestão de permissões).

Uma vez criada, uma mensagem não pode ser alterada (mas pode ser anexada a outras mensagens).

As organizações não podem enviar mensagens (mas podem recebê-las).

1.5.1 Envio de mensagens

Esta acção associa um identificador único à mensagem enviada.

Todas as mensagens enviadas são guardadas na caixa de saída (isto implica que, quando o agente envia uma mensagem para si próprio, fica com referências à mensagem, tanto na caixa de saída, como na de entrada). As mensagens nunca podem ser removidas da caixa de saída, mas podem ser removidas das caixas de entrada dos destinatários pelos próprios.

O envio de mensagens bem formadas é sempre possível e apenas possível por parte de agentes individuais. A recepção de mensagens depende das permissões estabelecidas (ver situações de erro abaixo). Mensagens provenientes do agente `0` são sempre entregues, independentemente de quaisquer restrições definidas pelos agentes.

Em geral, caso uma mensagem não possa ser entregue ao receptor, é gerada automaticamente uma mensagem de erro (à qual é anexada a mensagem não entregue), que é devolvida ao emissor. Estas mensagens são sempre provenientes do indentificador `0`.

Se, ao enviar a mensagem, houver identificadores de destino não admissíveis (e.g. bloqueados) ou inexistentes, o sistema envia de volta ao remetente uma mensagem (por identificador), indicando no assunto o problema. No primeiro caso, o assunto é `msgRejected()`. No segundo caso, o assunto é `unknownDestination()`. A mensagem original (sem anexos) é anexada à mensagem de erro.

Se, ao enviar a mensagem, houver identificadores de anexos não admissíveis (e.g. protegidos) ou inexistentes, o sistema envia de volta ao remetente uma mensagem (por identificador), indicando no assunto o problema. No primeiro caso, o assunto é `couldNotAttach()`. No segundo caso, o assunto é `unknownAttachment()`. A mensagem original (sem anexos) é anexada à mensagem de erro.

O código descrito acima não faz parte da interface textual e encontra-se definido no pacote **sonet-core-support**.

1.5.2 Respostas a mensagens

Esta acção associa um identificador único à mensagem enviada.

As respostas a mensagens incluem sempre a mensagem original como anexo. As respostas podem conter outros anexos (como qualquer mensagem), mas a mensagem original é sempre o primeiro.

O assunto da mensagem a enviar é o da mensagem original, precedido de `inReply()` (a cadeia de 4 caracteres `Re:␣`).

O código descrito acima não faz parte da interface textual e encontra-se definido no pacote **sonet-core-support**.

1.5.3 Reenvio de mensagens

Esta acção associa um identificador único à mensagem enviada.

Mensagens de reenvio incluem sempre a mensagem original como anexo. Os reenvios podem conter outros anexos (como qualquer mensagem), mas a mensagem original é sempre o primeiro.

O assunto da mensagem a enviar é o da mensagem original, precedido de `forwarded()` (a cadeia de 5 caracteres `Fwd:␣`).

O código descrito acima não faz parte da interface textual e encontra-se definido no pacote **sonet-core-support**.

1.6 Operações sobre ligações

Podem realizar-se as seguintes operações relativas a ligações: (i) visualizar; (ii) solicitar e aceitar; e, (iii) gerir permissões.

Quando um agente pede uma ligação, o seu lado da ligação fica aberto ao agente de destino (podendo ser, posteriormente, bloqueado), mesmo que o agente de destino não aceite a ligação.

Caso o agente de destino não aceite pedidos de ligação, é enviada uma mensagem (sem anexos) ao agente que origina o pedido, em que o assunto é o identificador de destino da ligação e o texto `unavailableId()`.

Caso o agente solicite uma ligação a um agente inexistente, é enviada uma mensagem (sem anexos) ao agente que pede a ligação, em que o assunto é o identificador de destino da ligação e o texto `unknownId()`.

Caso uma ligação seja aceite por um agente, é enviada uma mensagem (sem anexos) ao agente que pediu a ligação, com o assunto contendo o identificador do agente que aprovou a ligação e o texto `connectionApproved()`.

As organizações não podem solicitar ligações. Pedidos de ligação recebidos por uma organização são automaticamente aceites.

O bloqueio de uma ligação existente não implica o fim da ligação: será reactivada quando o bloqueio for removido.

O código descrito acima não faz parte da interface textual e encontra-se definido no pacote **sonet-core-support**.

1.7 Pesquisas

Deve ser possível efectuar pesquisas sob vários critérios e sobre as diferentes entidades geridas, e.g., ver todas as mensagens, ver todas as mensagens com anexos, ver agentes sem mensagens, ver agentes sem ligações, ver agentes sem publicações.

Deve ser possível introduzir novos métodos de pesquisa com um impacto mínimo na implementação desenvolvida.

2 Interacção com o Utilizador

As aplicações de gestão e de navegação utilizam a mesma base de código (o núcleo da aplicação), mas possuem dois conjuntos de menus independentes: o menu de gestão (§3) e o menu de navegação (§4). A aplicação de gestão pode ser iniciada sobre uma rede vazia (neste caso, todos os objectos devem ser criados através do menu de gestão). Outros objectos (associados a agentes), poderão ser mais tarde criados através da aplicação de navegação.

As excepções relacionadas com a interacção, excepto se indicado, são subclasses de `ist.po.ui.DialogException`, são lançadas pelos comandos e tratadas pela classe `ist.po.ui.Menu`. Note-se que, além das excepções descritas, é possível a definição de outras. As novas excepções não devem, no entanto, substituir as fornecidas nos casos descritos por este enunciado.

As operações de leitura (de dados fornecidos pelo utilizador) e escrita (para apresentação de resultados ao utilizador) **têm** de ser realizadas através do objecto `ist.po.ui.Dialog.IO`, utilizando-se as mensagens descritas (todas as mensagens são produzidas

através de chamadas a métodos). Uma lista completa das classes disponibilizadas pode ser obtida nas bibliotecas **po-uilib** e **sonet-textui-support** (material de apoio). As mensagens são apenas possíveis na interface (e nunca no núcleo da aplicação) e, em geral, não podem ser definidas novas. Casos potencialmente omissos devem ser esclarecidos antes de qualquer implementação.

A apresentação de listas faz-se por ordem crescente do critério de apresentação. Quando a chave é uma cadeia de caracteres, a ordem deve ser lexicográfica (UTF-8), não havendo distinção entre maiúsculas e minúsculas (§1.2).

Nas interacções descritas abaixo, os erros são comunicados tão cedo quanto possível. O pedido de um identificador é repetido sempre que a resposta não tiver o formato correcto. O formato correcto não é, contudo, uma garantia de que o identificador seja válido no contexto de utilização. Quando for pedida uma lista de identificadores, estes são verificados por ordem, apenas depois de lida toda a lista (excepto onde indicado).

O tipo de letra `fixo` indica um literal; o símbolo ␣ indica um espaço; e o tipo de letra *itálico* indica uma parte variável.

3 Aplicação de Gestão

As entradas do menu estão definidas em `sonet.textui.manager.MenuEntry`. Os métodos correspondentes às mensagens de diálogo estão definidos em `sonet.textui.manager.Message`.

Este menu permite realizar operações de leitura e escrita básicas e aceder a submenus que contêm a restante funcionalidade. A lista completa é a seguinte (omite-se a opção **Sair**, uma vez que é implementada automaticamente por todos os menus): **Criar**, **Abrir**, **Guardar**, **Guardar como...**, **Gestão de Agentes** (§3.2) e **Consultas** (§3.3). As secções abaixo descrevem pormenorizadamente as acções associadas a estas opções.

3.1 Manipulação de Ficheiros

O estado da aplicação pode ser guardado em ficheiros, para posterior recuperação: utiliza-se a capacidade de serialização do Java (implementações de `java.io.Serializable`). São definidas as operações básicas para manipulação de ficheiros: criação de novo ficheiro ("criar"), abertura de ficheiro existente ("abrir"), e salvaguarda de ficheiro aberto ("guardar"). Devem ser tratadas todas as excepções relativas à manipulação de ficheiros. Note-se que é sempre possível trabalhar com a aplicação, mesmo que não tenha sido carregado nenhum ficheiro. A funcionalidade de cada operação é a seguinte:

Criar – Criação de novo ficheiro.

Abrir – Abertura e eventual utilização de um ficheiro existente (previamente guardado). O sistema pede o nome do ficheiro a abrir (`openFile()`): caso não exista, o sistema limita-se a comunicar o erro (mensagem `fileNotFound()`).

Guardar – Salvaguarda das alterações desde a abertura do ficheiro associado à aplicação. Caso não haja nenhum ficheiro associado, deve ser perguntado ao operador o nome do ficheiro a utilizar. Esta interacção realiza-se através do método `newSaveAs()`. Não é executada nenhuma acção se não existirem alterações desde a última salvaguarda.

Guardar como... – Esta opção permite associar/mudar a associação de um ficheiro à aplicação (além de guardar os dados nesse ficheiro). Esta interacção utiliza o método `saveAs()`.

Apenas é possível trabalhar com um ficheiro de cada vez. Assim, sempre que se abandona um ficheiro com modificações não salvaguardadas, por exemplo, porque se cria outro, deve ser perguntado ao operador se deseja guardar a informação actual:

1. Se houver alterações, então deve-se perguntar ao operador se deseja guardar o ficheiro antes de sair: esta operação utiliza a mensagem `saveBeforeExit()` (a resposta é obtida invocando `readBoolean()`). Caso a resposta seja afirmativa, então deve-se guardar a informação.
2. Se não existir nenhum ficheiro associado, então deve-se ainda perguntar ao operador qual o nome a utilizar para guardar a informação, através da mensagem `newSaveAs()`.

Note-se que parte deste procedimento corresponde à opção Guardar.

3.2 Menu de Agentes

As entradas do menu estão definidas em `sonet.textui.agents.MenuEntry`. As mensagens de diálogo estão definidas em `sonet.textui.agents.Message`. As excepções a lançar pelos comandos estão definidas em `sonet.textui.agents`.

Sempre que for pedido o identificador do agente (`reqKey()`) e o identificador não existir (excepto no processo de registo), o comando deve lançar `UnknowKeyException`. As listas de identificadores são pedidas com `reqKeys()`.

3.2.1 Visualizar todos os agentes

O formato de apresentação de cada agente é o seguinte:

type | *id* | *name* | *email* | *#pub* | *#imsg* | *#omsg* | *#conn* | *active?*

Os valores para o campo *type* são `PERSON` (`typePerson()`), ou `ORGANIZATION` (`typeOrganization()`), e os valores para o campo *active?* são `ACTIVE` (`agentActive()`) e `INACTIVE` (`agentInactive()`). Estes métodos são fornecidos apenas por conveniência e não implicam nenhuma opção relativamente ao desenho da aplicação. *#pub*, *#imsg*, *#omsg* e *#conn* indicam, respectivamente, o número de publicações, mensagens (recebidas e enviadas) e ligações.

3.2.2 Registar novo agente

É pedido o tipo do agente (`reqType()`) (devendo ser dada uma resposta como indicado para o campo *type* em §3.2.1), o nome do agente (`reqName()`), o endereço de correio electrónico (`reqEmail()`) e o número de telefone (`reqPhone()`).

Se o tipo indicado for desconhecido, repete-se a pergunta até se obter uma resposta correcta.

Esta acção associa um identificador único ao agente criado.

3.2.3 Desactivar um agente

É pedido o identificador do agente (§3.2). O comando deve lançar `AgentIsInactiveException` para agentes inactivos.

3.2.4 Reactivar um agente

É pedido o identificador do agente (§3.2). O comando deve lançar `AgentIsActiveException` para agentes activos.

3.3 Menu de Consultas

As entradas do menu estão definidas em `sonet.textui.search.MenuEntry`. As mensagens estão definidas em `sonet.textui.search.Message`.

Sempre que for feita uma consulta e nenhuma entidade satisfazer as condições associadas ao pedido, nada deve ser impresso.

Entrada do menu	Acção
Ver todas as mensagens	As mensagens devem ser apresentadas como indicado em §4.4.1.
Ver agentes sem mensagens	Os agentes devem ser apresentados como indicado em §3.2.1.

4 Aplicação de Navegação

A aplicação de navegação solicita o identificador de um agente (`sonet.textui.browser.Message.login()`). É possível utilizar o agente `0`: neste caso, as restrições de acesso não se aplicam. Se o identificador corresponder a uma organização, algumas operações não estão disponíveis (ver abaixo). Depois de validado, o identificador de agente indicado fica activo e os direitos de utilização dos menus subsequentes são a ele relativos.

De seguida é aberto o ficheiro `sonet.dat` que contém os dados da rede. Note-se que os dados são reescritos neste ficheiro quando a aplicação termina. Assume-se, por simplicidade (testes automáticos), que o ficheiro indicado existe sempre e que contém dados correctos (contudo, uma boa implementação deveria verificar os problemas de abertura e leitura dos objectos).

Após o diálogo inicial, é aberto o menu de inspecção de um agente (§4.1).

4.1 Menu de navegação

As entradas do menu estão definidas em `sonet.textui.browser.MenuEntry`.

É pedido o identificador de um agente (§3.2). De seguida, é aberto o menu de inspecção para o agente seleccionado (este menu permite gerir a informação que o agente tem na rede). Note-se que, se o identificador indicado nesta opção for igual ao de "login", o agente está a inspeccionar os seus próprios dados e que apenas neste caso poderá fazer algumas operações.

O símbolo **[A]** significa que uma operação apenas é executada pelo agente sobre si próprio e não pode ser executada sobre outros.

O símbolo **[P]** significa que uma operação pode ser executada pelo agente sobre outros (incluindo o próprio), desde que exista uma permissão activa.

4.2 Inspecção de um Agente

As entradas do menu estão definidas em `sonet.textui.agent.MenuEntry`. As mensagens de diálogo estão definidas em `sonet.textui.agent.Message`.

O título do menu é construído com o método `menuTitle()` (recebe o identificador do agente) e não com a entrada `TITLE`.

4.2.1 Visualizar propriedades [P]

Apresentação idêntica à de §3.2.1, mas apenas disponível se a permissão de visualização de perfil estiver activa.

4.2.2 Editar perfil do agente [A]

As acções desempenhadas aqui são semelhantes às de registo (§3.2.2), mas não se pode alterar o tipo de agente, nem o estado de actividade. É possível alterar o perfil de um agente inactivo.

É pedido o nome do agente (`reqName()`), o endereço (`reqEmail()`) e o número de telefone (`reqPhone()`). Estas mensagens estão definidas em `sonet.textui.agents.Message`.

4.2.3 Proteger perfil do agente [A]

É pedida uma lista de identificadores de agentes (separada por vírgulas) (§3.2). Os identificadores indicados não podem visualizar nenhuma informação sobre o agente (incluindo publicações).

4.2.4 Desproteger perfil do agente [A]

É pedida uma lista de identificadores de agentes (separada por vírgulas) (§3.2). Os identificadores indicados podem visualizar informação do perfil do agente (mas não necessariamente publicações).

4.2.5 Menu de publicações [P]

Abre o menu de gestão de publicações do agente inspeccionado (§4.3).

4.2.6 Menu de mensagens [A]

Abre o menu de gestão de mensagens do agente (§4.4).

4.2.7 Menu de ligações [P]

Abre o menu de gestão de ligações do agente inspeccionado (§4.5).

4.3 Gestão de Publicações

As entradas do menu estão definidas em `sonet.textui.publication.MenuEntry`. As mensagens de diálogo estão definidas em `sonet.textui.publication.Message`. As excepções a lançar pelos comandos estão definidas em `sonet.textui.publication`.

Deve ser lançada `UnknowKeyException`, sempre que for pedido o identificador da publicação (`reqKey()`) e o identificador não existir (excepto no processo de registo).

4.3.1 Listar publicações [P]

O formato de apresentação do cabeçalho de cada publicação na lista é o seguinte:

type | *id* | *rating+* | *rating-* | *#comments*

Os valores para o campo *type* são `NOTE` (`typeNote()`), `URI` (`typeURI()`) e `IMAGE` (`typeImage()`). Estes métodos são fornecidos apenas por conveniência e não implicam nenhuma opção relativamente ao desenho da aplicação. *rating+* e *rating-* indicam respectivamente os somatórios de votos positivos e negativos, e *#comments* indica o número de comentários.

4.3.2 Mostrar publicação [P]

É pedido o identificador da publicação a visualizar (§4.3), sendo apresentado o cabeçalho descrito em §4.3.1. De seguida, são apresentados os conteúdos da publicação: para notas textuais, é apresentado o texto; para imagens, é apresentada a "image"; e, finalmente, para URIs, é apresentado o valor correspondente à URI. Finalmente, são apresentados os comentários (por ordem de registo).

4.3.3 Registar nova publicação [A]

É pedido o tipo do publicação (`reqType()`) (devendo ser dada uma resposta como indicado para o campo *type* em §4.3.1), a legenda da publicação (`reqLegend()`). Para notas textuais, é ainda pedido o texto (`reqNoteText()`); para imagens, é pedida a "image" (`reqImgContents()`); e, finalmente, para URIs, é pedido o valor correspondente (`reqURI()`).

Se o tipo indicado for desconhecido, repete-se a pergunta até se obter uma resposta correcta.

Esta acção associa um identificador único à publicação criada.

4.3.4 Pontuar publicação [P]

São pedidos o identificador da publicação (§4.3) e o valor numérico inteiro (positivo ou negativo) para pontuar a publicação (`reqRating()`). O comando deve lançar `RatingDeniedException` se a publicação estiver inacessível ao agente que pontua.

4.3.5 Comentar publicação [P]

São pedidos o identificador da publicação (§4.3) e o comentário a anexar à publicação (`reqComment()`). O comando deve lançar `CommentDeniedException` se a publicação estiver inacessível ao agente que comenta.

4.3.6 Proteger publicação [A]

São pedidos o identificador da publicação (§4.3) e uma lista de identificadores de agentes (separada por vírgulas) (§3.2) aos quais será bloquado o acesso.

4.3.7 Proteger todas as publicações actuais do agente [A]

É pedida uma lista de identificadores de agentes (separada por vírgulas) (§3.2) aos quais será bloquado o acesso.

4.3.8 Desproteger publicação [A]

São pedidos o identificador de uma publicação (§4.3) e uma lista de identificadores de agentes (separada por vírgulas) (§3.2) aos quais será concedido acesso.

4.3.9 Desproteger todas as publicações actuais do agente [A]

É pedida uma lista de identificadores de agentes (separada por vírgulas) (§3.2) aos quais será concedido acesso.

4.4 Menu de Mensagens

As entradas do menu estão definidas em `sonet.textui.message.MenuEntry`. Os métodos correspondentes às mensagens de diálogo estão definidos em `sonet.textui.message.Message` (excepto no processo de registo). As excepções a lançar pelos comandos estão definidas em `sonet.textui.message`.

Excepto quando indicado, quando for pedido o identificador de uma mensagem (`reqKey()`) e o identificador não existir, deve ser lançada `UnknownKeyException`.

As operações devem respeitar a semântica descrita em §1.5.

4.4.1 Visualizar mensagens recebidas (caixa de entrada)

O formato de apresentação do cabeçalho de cada mensagem na lista é o seguinte:

`MESSAGE` | *id* | *from* | *to* | *subject* | *#attachments*

Os campos *from* e *to* são os identificadores dos agentes envolvidos. O campo *to* pode conter mais de um agente (neste caso, usam-se vírgulas para separar os identificadores correspondentes).

4.4.2 Visualizar mensagens enviadas (caixa de saída)

Apresenta as mensagens enviadas, utilizado o formato apresentado em §4.4.1.

4.4.3 Visualizar mensagem

É pedido o identificador da mensagem (§4.4), sendo apresentado o cabeçalho da mensagem (§4.4.1), seguido do texto da mensagem. Se existirem anexos, são apresentados por ordem, depois do texto. O formato de apresentação dos anexos depende de cada anexo (§4.4.3, para mensagens; §4.3.2, para publicações).

4.4.4 Enviar mensagem

A semântica de base é a descrita em §1.5.1.

São pedidos os identificadores dos destinatários (lista separada por vírgulas) (§3.2). Se a lista de identificadores for vazia, não é enviada nenhuma mensagem e a acção é interrompida: o comando deve lançar `NoDestinationException`.

De seguida, são pedidos o assunto (`reqSubject()`) e o texto da mensagem (`reqText()`): devem ser lidas todas as linhas de texto até aparecer uma que corresponde a um único ponto (“.”). A linha com o ponto faz parte da mensagem.

De seguida, são pedidos os identificadores de eventuais anexos (`reqKeys()`) (publicações ou mensagens; lista separada por vírgulas).

4.4.5 Responder a mensagem

A semântica de base é a descrita em §1.5.2.

É realizada a operação correspondente à visualização de uma mensagem (§4.4.3).

De seguida, são pedidos o texto da mensagem e os anexos (tal como indicado em §4.4.4).

4.4.6 Reenviar mensagem

A semântica de base é a descrita em §1.5.3.

É realizada a operação correspondente à visualização de uma mensagem (§4.4.3).

De seguida, são pedidos os identificadores dos destinatários (lista separada por vírgulas) (§3.2). Se a lista de identificadores for vazia, não é enviada nenhuma mensagem e a acção é interrompida: o comando deve lançar `NoDestinationException`.

De seguida, são pedidos o texto da mensagem e os anexos (tal como indicado em §4.4.4).

4.4.7 Remover mensagem

É realizada a operação correspondente à visualização de uma mensagem (§4.4.3).

De seguida é pedida a confirmação de remoção da mensagem da caixa de entrada (`reqRemoveMessage()`). Em caso afirmativo, é apresentada a mensagem `messageRemoved()`; em caso negativo, não é realizada nenhuma acção.

4.4.8 Autorizar mensagens

É pedida uma lista de identificadores de agentes (separada por vírgulas) (§3.2). Os identificadores indicados ficam autorizados a enviar mensagens ao utilizador, e podem reenviar mensagens que tenham tido origem nele.

4.4.9 Bloquear mensagens

É pedida uma lista de identificadores de agentes (separada por vírgulas) (§3.2). Os identificadores indicados ficam impedidos de enviar mensagens ao utilizador, e de reenviar mensagens que tenham tido origem nele.

4.5 Gestão de Ligações

As entradas do menu estão definidas em `sonet.textui.connection.MenuEntry`. Os métodos correspondentes às mensagens de diálogo estão definidos em `sonet.textui.connection.Message` (excepto no processo de registo).

4.5.1 Listar ligações [P]

São apresentadas todas as ligações do agente, no seguinte formato:

`CONNECTION` | *id* | *active?*

O campo *id* é o identificador do destino na lista do agente de origem (define a ordenação de visualização). Quando a ligação está estabelecida, o campo *active?* tem o valor `1` quando é possível comunicar com o outro agente. Se o outro agente ainda não aceitou a ligação, tem o valor `0`.

Note-se que não é possível saber, por esta via, se uma ligação está autorizada pelo outro agente.

4.5.2 Pedido de ligação [A]

É pedida uma lista de identificadores de agentes (separada por vírgulas) (§4.4), sendo enviados pedidos de ligação (não são mensagens) aos identificadores indicados. Se um identificador de destino corresponder a uma organização, a ligação é imediatamente aceite.

4.5.3 Aceitar ligações [A]

É pedida uma lista de identificadores de agentes (separada por vírgulas) (§4.4). Se os identificadores forem inválidos ou não tiverem pedido ligações, não é realizada nenhuma acção.

5 Importação de Dados Textuais

A introdução de dados através da aplicação de gestão permite criar entidades na rede. No entanto, o processo é moroso e é possível substituí-lo pela criação de objectos a partir de uma descrição textual (apresentada abaixo; este exemplo está presente no ficheiro `data.txt`). Note-se que estes dados textuais são fornecidos apenas como forma de tornar cómoda a inicialização e nunca são produzidos pela aplicação (nem mesmo para salvaguardar o seu estado para execuções futuras).

No processamento destes dados, assume-se que não há entradas mal formadas (em particular, todos os identificadores referidos já estão registados). Sugere-se a utilização do método `String.split`, para dividir uma cadeia de caracteres em campos.

```
PERSON|1|Luke␣Skywalker|wimp@resistance.org|123-456-789
PERSON|2|Darth␣Vader|vader@imperial.net|911
PERSON|3|Master␣Yoda|yoda@jedi.edu|1
PERSON|4|Montgomery␣Scott|scotty@starfleet.net|987-654-321
PERSON|5|James␣T.␣Kirk|kirk@starfleet.net|987-654-321
PERSON|6|Jean-Luc␣Picard|picard@starfleet.net|987-123-654
ORGANIZATION|7|Google|noreply@google.com|0
ORGANIZATION|8|Melkor␣&␣Sauron's|recruiting@utumno.com|999
ORGANIZATION|9|Starfleet|info@starfleet.net|999
MESSAGE|10|1|3|cities|look␣at␣this␣http://bit.ly/p0XoDW
MESSAGE|11|2|1|we␣have␣to␣talk|Luke,␣I'm␣your␣father!
MESSAGE|12|1|2|Re:␣we␣have␣to␣talk|Noooooooooooo!!!!
MESSAGE|13|5|4|Teleporter|Beam␣me␣up,␣Scotty!
MESSAGE|14|6|4|Teleporter|Beam␣me␣up,␣Scotty!
NOTE|15|3|Cookies|Mix␣everything␣and␣apply␣a␣little␣Force␣for␣30␣seconds
NOTE|16|4|Dilithium␣crystals|Remember␣never␣to␣put␣them␣in␣the␣freezer
URI|17|1|Angry␣Ewoks|http://jedi.edu/ewoks/
URI|18|2|Imperial␣Bank|http://bank.imperial.net/
IMAGE|19|1|Tatooine|[desert␣scene␣with␣two␣suns␣setting]
CONNECTION|1|2|0|1
CONNECTION|1|3|1|1
CONNECTION|4|5|1|1
CONNECTION|4|6|1|1
```

Os formatos são os seguintes. Note-se que algumas das possibilidades do modelo não estão previstas nestes formatos (e.g., anexos em mensagens). Isto é uma mera simplificação do formato da entrada, mas não do modelo de objectos.

`PERSON` | *id* | *name* | *email* | *phone*
`ORGANIZATION` | *id* | *name* | *email* | *phone*
`MESSAGE` | *id* | *from* | *to* | *subject* | *body*
`NOTE` | *id* | *owner* | *title* | *text*
`IMAGE` | *id* | *owner* | *description* | *content*
`URI` | *id* | *owner* | *description* | *uri*
`CONNECTION` | *id1* | *id2* | *id1-accepting?* | *id2-accepting?*

6 Execução dos Programas e Testes Automáticos

Usando os ficheiros `data.txt`, `input.dat` e `output.dat` (a disponibilizar) é possível verificar automaticamente se o programa produz o resultado esperado para um exemplo. Supondo que os ficheiros `input.dat` e `output.dat` estão no directório onde é dado o comando de execução e que a localização do pacote `sonet` é possível com a definição apropriada da variável `CLASSPATH` (ou da opção equivalente `-cp`), se for dado o comando abaixo, deve ser produzido um ficheiro de saída `output1.dat`, que deve ser igual ao ficheiro `output.dat` (o ponto de entrada da aplicação a desenvolver é o método `sonet.textui.Manager.main`):

```
java -DImport=data.txt -Din=input.dat -Dout=output1.dat sonet.textui.Manager
java                   -Din=input.dat -Dout=output1.dat sonet.textui.Browser
```

Os ficheiros de saída (esperada e obtida) devem ser iguais em caso de sucesso. A comparação pode ser feita com o comando:

```
diff -b output.dat output1.dat
```

Este comando não deve produzir qualquer resultado quando os ficheiros são iguais. Note-se, contudo, que este teste não garante o correcto funcionamento do código desenvolvido, apenas verificando algumas funcionalidades.