



Análise e Desenho de Software

3ª fase do Trabalho 2022/2023

Enunciado

Pretende-se com este trabalho que os alunos exercitem a compreensão dos artefactos UML criados durante o processo de desenvolvimento da aplicação AgentADS e dos padrões de desenho abordados na disciplina, implementando em Java esses artefactos e técnicas. Para isso, são apresentados em três documentos fornecidos:

- os SSDs dos casos de uso a implementar;
- os diagramas de interação das várias operações do sistema desses casos de uso; e
- o diagrama de classes do desenho.

Os alunos deverão implementar as classes da camada do domínio (*domain*). As classes que constituem uma possível camada de *interface* com o utilizador são dadas. É dado também um pacote de classes do domínio que têm a ver com verificadores de “força” de *passwords*.

Os casos de uso a implementar são:

1. Login de Utilizador
2. Criar novo Agente
3. Consultar informação de Agente
4. Adicionar documento para Agente
5. Ler documento de Agente
6. Registar indisponibilidade de Agente
7. Criar nova Missão
8. Consultar informação de Missão

Os casos de uso “Criar novo Agente”, “Criar nova Missão”, “Registar indisponibilidade de Agente” estão ligeiramente diferentes dos apresentados nas 2 primeiras fases.

- Considerou-se que a manipulação de info sobre agentes e missões carecia de maior segurança pois qualquer pessoa que conseguisse aceder à área de um utilizador poderia, por ex., criar uma missão, registar a indisponibilidade de um agente, etc, bastando para isso saber o nome de código desse agente (o que consegue, por ex., acedendo à info sobre uma dada missão, o qual conseguirá por ex., fazendo uma procura de missões por uma dada característica).
- Assim sendo, acrescentou-se o pedido da chave de acesso das missões e agentes, sempre que um utilizador quer aceder à sua informação (consulta e/ou alteração).
- Como no agente a *access key* representava a chave de codificação/descodificação (agora chamada de *coding key*), acrescentámos uma nova chave que o utilizador terá que dar sempre que manipular a informação do agente (aos utilizadores não criadores de missão ou agente, mas que a eles têm acesso, serão dadas a conhecer essas chaves pessoalmente, “de boca para ouvido”, para depois poderem fornecê-las quando tal lhes for solicitado).

Pacote *domain*

O pacote *domain*, a implementar pelos alunos, deve incluir todas as classes, *interfaces* e enumerados descritos no diagrama de classes do desenho (DCD), apresentado no documento *DiagClassesAgentADS.pdf*. Os métodos deverão implementar os desenhos dos diagramas de interação (IDs), apresentados no documento *IDsAgentADS.pdf*.

Para maior legibilidade, no DCD não se mostram os métodos de cada classe. Segue-se, para algumas classes/enumerados/interfaces, uma descrição mais pormenorizada para colmatar algumas dúvidas que possam ficar após inspeção do DCD e IDs.

- o enumerado **DocNature** define valores correspondentes aos tipos de documentos existentes na OCT. Para já teremos **TO_AGENT**, **FROM_AGENT** e **INTERNAL**. Neste enumerado, cada valor deverá ter uma *string* associada que deverá poder ser acedida através do método
 - **String getword()** que retornará os valores “ToAg”, “FrAg” e “Intr”, respetivamente.
 - o enumerado **Availability** define valores de *availability* de um agente. Para já teremos **AVAILABLE**, **ARRESTED**, **DESERTED**, **ILL**, **DEAD** e **RETIRED**.
 - a classe **Document** deve também redefinir os métodos **toString** e **clone**.
 - a classe **UniqueIdentifierFactory** representa uma fábrica de referências únicas para os documentos. É um *Singleton* que oferece os seguintes métodos de instância:
 - **IndicatorFactory getInstance()** que devolve a única instância da classe;
 - **String getIdentifier(DocNature dn)** que devolve uma referência única para um documento do tipo dado por **dn**; esta referência deve ser composta pela palavra correspondente ao tipo **dn** seguida de um número sequencial (p.ex., se a última referência criada pela fábrica para documentos do tipo **TO_AGENT** tiver sido “ToAg3”, a próxima para este tipo deverá ser “ToAg4”).
 - o interface **ICodifierStrategy**, define estratégias de codificação de documentos. Contém os métodos:
 - **String getName()** que devolve o nome da estratégia de codificação (ou codificador);
 - **Iterable<String> code (String key, Iterable<String> text)** que devolve as linhas de texto resultantes de codificar o texto **text** usando a chave **key**;
 - **Iterable<String> decode (String key, Iterable<String> text)** que devolve as linhas de texto resultantes de decodificar o texto **text** usando a chave **key**;
 - a classe abstrata **AbstractCodifierStrategy** implementa o método **String getName()**.
 - a classe **IdentityCodifierStrategy** implementa os métodos **code** e **decode** de forma a que os textos codificados/descodificados sejam iguais aos dados pelo parâmetro **text**; o seu nome deverá ser “Identity”.
 - a classe **Round13CodifierStrategy** implementa os métodos **code** e **decode** de forma a que cada letra seja codificada na letra que fica 13 posições à direita no alfabeto (de 26 letras); considera-se que a seguir ao “z” vem o “a” e que a seguir ao “Z” vem o “A”; a decodificação é feita exatamente da mesma maneira; o seu nome deverá ser “Round13”.
 - a classe **VigenereCodifierStrategy** implementa os métodos **code** e **decode** de forma a obter a codificação/descodificação definidas pela cifra Vigenère (ver https://en.wikipedia.org/wiki/Vigenère_cipher) ; o seu nome deverá ser “Vigenere”.
 - o interface **Alert** define tipos de alertas que se querem observados. Contém dois métodos:
 - **String getSubjectName()** que devolve o nome do sujeito que provocou o alerta;
 - **String getMessage()** que devolve a mensagem associada ao alerta;
 - as classes **AgentUnavailableAlert**, **AgentInfoSearchedAlert** e **MissionInfoSearchedAlert**, definem tipos específicos de alerta, para usar:
 - **AgentUnavailableAlert**: quando um agente se torna indisponível (mais precisamente, quando o seu método **becomeUnavailable()** é chamado),
 - **AgentInfoSearchedAlert**: quando a informação de um agente é consultada (mais precisamente, quando algum dos seus métodos **codifierName()**, **documentReferences()**, ou **decodifiedDocText(String ref)** é chamado),
 - **MissionInfoSearchedAlert**: quando a informação de uma missão é consultada (mais precisamente, quando seu o método **getParticipants()** é chamado).
- Oferecem construtores com dois parâmetros: **String subject** que define o nome do agente ou da missão que provocou o alerta e **String message** que define a mensagem que fica associada ao alerta.
- na classe **Agent**
 - o método **String codifierName()**, que devolve o nome do codificador do agente, também deve notificar os *observers* do agente, usando para isso uma instância de **AgentInfoSearchedAlert** com a mensagem “Agent XXX codifier name was searched” onde XXX é o nome de código deste agente.;
 - o método **Iterable<String> documentReferences()**, que devolve as referências dos documentos

“de” e “para” o agente, também deve notificar os *observers* do agente, usando para isso uma instância de **AgentInfoSearchedAlert** com a mensagem “Agent XXX doc references were searched” onde XXX é o nome de código deste agente;

- o método **Iterable<String> decodedDocText(String ref)**, que, se o documento de referência **ref** é um dos documentos deste agente, devolve o texto desse documento já decodificado, também deve notificar os *observers* do agente, usando para isso uma instância de **AgentInfoSearchedAlert** com a mensagem “Agent XXX document YYY was searched” onde XXX é o nome de código deste agente e YYY é **ref**;
- de forma semelhante, o método **void becomeUnavailable(String unavail)**, que regista a indisponibilidade do agente para o valor do enumerado **Availability** correspondente a **unavail** (método **valueOf** do enumerado), deve notificar os *observers* do agente, usando para isso uma instância de **AgentUnavailableAlert** com a mensagem “Agent XXX became YYY” onde XXX é o nome de código deste agente e YYY é **unavail**;
- o método **toString** deve ser redefinido (nele deve ser usado o método **toString** da classe **Document**).

▪ na classe **Mission**

- o método **Iterable<String> getParticipants()**, que devolve os nomes de código dos agentes que participam na missão (não inclui o agente responsável), deve notificar os *observers* da missão, usando para isso uma instância de **MissionInfoSearchedAlert** com a mensagem “Mission XXX info was searched” onde XXX é o nome de código da missão;
- no método **void propertyChange(PropertyChangeEvent ev)**, método definido no *interface* **java.beans.PropertyChangeListener**, se **ev.getNewValue()** for do tipo **AgentUnavailableAlert**, deve-se remover da missão o agente cujo nome é dado por **(Alert)ev.getNewValue().getSubjectName()**; isto deve acontecer seja esse agente um participante ou o responsável da missão; não esquecer de quando se remove o agente, pedir-lhe que apague esta missão da sua lista de *observers*;
- o método **toString** deve ser redefinido (nele deve ser usado o método **toString** da classe **Agent**).

▪ na classe **User**,

- no método **void propertyChange(PropertyChangeEvent ev)**, método definido no *interface* **java.beans.PropertyChangeListener**, se **ev.getNewValue()** for do tipo **AgentUnavailableAlert**, **AgentInfoSearchedAlert** ou **MissionInfoSearchedAlert** ele deve ser adicionado ao conjunto de alertas deste user;
- o método **toString** deve ser redefinido (nele deve ser usado o método **toString** das classes **Agent** e **Mission** e o método **getMessage()** dos alertas).

▪ a classe **CodifierFactory** poderá ser feita à semelhança da classe **domain.verifiers.VerifierFactory**, ou usando a abordagem ilustrada nos slides 33 a 35 do ficheiro *TeosPadrõesGRASPeGOF.pdf*, acessível no Moodle.

▪ na classe **UserCatalog**: pelo facto de não ser implementado o caso de uso Criar Utilizador e para ser possível testar a aplicação AgentADS, o construtor desta classe deverá criar e guardar 3 users: a “Mary”, o “Peter” e o “John Snow” cujas passwords serão, respetivamente, “123”, “ABC” e “123ABC”.

▪ na classe **OCT**: deve ser implementado o método **String getUsersInfo(String name)** que devolve o resultado de invocar **toString()** sobre o user cujo nome é **name**; este método não faz parte da *interface* **domain.interfaces.IOCT** mas deve ser implementado para permitir inspecionar o estado dos users a partir da classe **TextClient**.

Pacote *domain.interfaces* (fornecido)

Contém as *interfaces* que o domínio torna visível a outros pacotes. Neste caso contém as *interfaces* correspondentes aos *handlers* dos casos de uso e a *interface* que define o Objeto Inicial (**IOCT**). Por exemplo, a aplicação GUI por nós fornecida está construída só com base neste pacote, dando liberdade a que a implementação do domínio possa ser feita livremente (desde que implemente todas estas *interfaces*). Um exemplo da aplicação do princípio *information hiding*.

Pacote *domain.handlers* (a fazer pelos alunos)

O pacote **domain.handlers** irá conter implementações dos *handlers* dos casos de uso, de acordo com os *interfaces* do pacote **domain.interfaces**. A classe **domain.OCT** (que podem ver no DCD) irá criar e devolver

instâncias destas classes (ver IDs das operações desta classe).

Pacote `domain.stubs` (fornecido)

É fornecido um pacote `domain.stubs` que contém uma implementação “fingida” dos *handlers* dos casos de uso de forma a que a aplicação GUI por nós fornecida funcione desde já, sem as classes do `domain` construídas. Estas implementações simplesmente imprimem os nomes das operações do caso de uso que vão sendo chamadas a partir da aplicação GUI. Serve para ilustrar quais os métodos invocados pela aplicação GUI e por que ordem são chamados (notem que a ordem respeita os SSDs de cada caso de uso).

Neste pacote é também fornecida uma implementação da classe `OCT` que devolve, a pedido, os *handlers* dos casos de uso. Nesta implementação de `OCT`, os *handlers* que são devolvidos são os deste pacote `domain.stubs`

Pacote `startup` (fornecido)

O pacote `startup` contém o código responsável pelo arranque da aplicação. Contém duas classes: a classe `Startup` que executa a aplicação GUI e a classe `TextClient` sem *input* do utilizador e com *output* para a consola.

- A classe `Startup` tem o método `static void main (String[] args)` que executa o caso de uso *Start-Up* (criação do Objeto Inicial), e cria também o objeto responsável pela interface gráfica com o utilizador.

Tal como vos é fornecida, a classe `Startup` cria um objeto do tipo `domain.stubs.OCT` para que a aplicação execute corretamente na situação em que as classes do pacote `domain` ainda não existem. A ideia é vocês criarem a classe `domain.OCT` que cria os verdadeiros *handlers* e alterarem na classe `Startup` a declaração `import domain.stubs.OCT` para `import domain.OCT`.

Sugerimos que criem inicialmente a classe `domain.OCT` exatamente igual à classe `domain.stubs.OCT` e que vão alterando a primeira à medida que vão construindo as classes necessárias aos vários casos de uso. Sempre que se torne possível testar um dado caso de uso `xxx`, devem substituir na classe `domain.OCT` a declaração `import domain.stubs.xxxHandler` por `import domain.xxxHandler`.

A interface gráfica utiliza as classes dos pacotes *view* e *controller* (ver mais à frente); estas estão construídas utilizando Java FX 20 e por isso, para conseguir que a classe `Startup` execute o interface gráfico, é necessário dar uma série de passos descritos mais à frente, na página 6.

- A classe `TextClient` tem um método `static void main (String[] args)` que permite testar a aplicação sem a intervenção do utilizador. Deverão usar esta classe para verem se os resultados da vossa implementação estão de acordo com os apresentados no exemplo no Anexo I (pág 8). Tal como vos é fornecida, a classe cria um objeto do tipo `domain.stubs.OCT` para que a aplicação execute corretamente na situação em que as classes do pacote `domain` ainda não existem.

Pacote `domain.verifiers` (fornecido)

O pacote `domain.verifiers` contém as classes `VerifierFactory`, `AbstractPwdStrengthAdapter`, `DummyAdapter`, `ExigentAdapter` e `EasyGoingAdapter` e o interface `IPwdStrengthAdapter`. Trata do carregamento dinâmico das classes que permitem a ligação a classes externas à aplicação e que fornecem o serviço de verificação da “força” de *passwords*. Embora (i) seja uma implementação exemplo e não faça ligação com classes externas reais e (ii) os alunos não tenham que fazer qualquer tipo de alteração às classes deste pacote, serve como um bom exemplo do carregamento dinâmico de classes e da implementação dos padrões GoF *Factory* (`VerifierFactory`), *Adapter* (`IPwdStrengthAdapter`, `AbstractPwdStrengthAdapter`, `DummyAdapter`, `ExigentAdapter` e `EasyGoingAdapter`) e *Singleton* (`VerifierFactory`).

Pacote `services` (fornecido)

O pacote `services` contém a classe `SessionManager` responsável por gerir a sessão dos utilizadores com o sistema. Esta é uma implementação exemplo e o seu único objetivo é ilustrar que alguns serviços podem ser dados por camadas que estão por baixo da camada de domínio. A persistência de informação (que não abordamos neste projeto) é outro exemplo de um serviço que pode ser oferecido à camada de domínio.

Pacotes *view* e *controller* (fornecidos)

Estes dois pacotes foram implementados por uma equipa diferente. Contêm as classes necessárias para a aplicação interagir com o utilizador nos casos de uso considerados. Foi desenvolvido seguindo o padrão

arquitetural *Model-View-Controller* e os controladores invocam os métodos dos *handlers* do pacote `domain`.

A interface gráfica de utilizador está construída utilizando Java FX 20 e, como tal, necessita do Java 17 ou posterior instalado nas vossas máquinas para correr o projeto. Ver na página 6 o que fazer para conseguir eliminar os erros destas classes e executar sem problemas.

Pode ver no anexo II (pág 16) um breve manual de utilizador da interface gráfica.

Ficheiro de texto `meeting.txt` (fornecido)

No projeto é também fornecido um ficheirinho de texto para ser usado na criação de um documento para o agente. É este ficheiro que é usado no exemplo da classe `startup.TextClient`.

Utilização de padrões de desenho

No desenho do modelo de classes apresentado, foram usados uma série de padrões de desenho (ver matérias dadas nas aulas teóricas e TPs). Seguem-se alguns padrões e exemplos de utilização neste projeto:

- **Factory.** Usamos fábricas para criar os verifiers, os codificadores e as referências para documentos;
- **Singleton.** As fábricas, por exemplo;
- **Iterator.** A linguagem Java tem suporte para este padrão, através dos interfaces `java.util.Iterator` `java.util.Iterable` e do ciclo `foreach`. Alguns dos métodos que devolvem listas de nomes (e.g., de users, de codificadores, de verifiers, etc) têm `Iterable` como tipo de retorno;
- **Façade.** As classes do domínio devem ser protegidas por uma fachada; os *handlers* dos casos de uso são casos particulares de objetos fachada; a classe `SessionManager` do pacote `services` é a fachada de um subsistema de classes e interfaces que gerem as sessões;
- **Observer.** Notificação entre `domain.Agent` e `domain.Mission` (observáveis) e `domain.User` (observador) e também entre `domain.Agent` (observável) e `domain.Mission` (observador);
- **Strategy.** A interface `domain.ICodifierStrategy` e a classe abstrata `domain.AbstractCodifierStrategy` “embrulham” os vários codificadores de documentos e as várias estratégias de codificação;
- **Adapter.** A interface `domain.IPwdStrengthAdapter` embrulha as várias implementações de adaptadores de verificadores de “força” de *passwords*;
- **Prototype.** Outro padrão que a linguagem Java já implementa nativamente. Notem que a classe `domain.verifiers.VerifierFactory` carrega dinamicamente classes (protótipos) para memória a partir dos quais cria objetos;
- **Protected Variations.** “Proteger” os elementos da aplicação contra as variações noutros elementos (objetos, sistemas, subsistemas), “embrulhando” o foco de instabilidade com uma *interface* (ou classe abstrata) e usando polimorfismo para usar várias implementações dessa interface. Vários dos padrões referidos acima (e.g., Strategy, Adapter, Factory) são variações mais específicas deste padrão;
- **Model-View-Controller.** Este padrão arquitetural promove o desenvolvimento de aplicações em camadas e é usado tipicamente em aplicações com *user interfaces* ricas; os pacotes `domain`, `view` e `controller` da nossa aplicação `AgentADS` seguem este padrão.

O que há a fazer

Produzir as classes, enumerados e *interfaces* Java mencionadas anteriormente e eventualmente outras classes ou métodos que considere relevantes. A relação de dependência entre as classes define naturalmente uma ordem de implementação: primeiro desenvolvem-se as classes sem dependências, e depois as que apenas dependem destas, e assim sucessivamente.

É preciso tomar em atenção que a solução de desenho descrita nos ficheiros anexos (*DiagClassesAgentADS.pdf* e *IDsAgentADS.pdf*) não é suficientemente detalhada para que não tenha de tomar ainda algumas decisões ao nível da implementação (por exemplo, a acessibilidade das classes e métodos, a fatorização de código através de definição e utilização de classes abstratas, a **redefinição apropriada** dos métodos herdados de *Object*). Devem ser tomadas as decisões que forem consideradas mais adequadas relativamente a tudo o que não estiver descrito.

Todas as classes **devem estar convenientemente documentadas**. Além dos comentários em *javadoc* (úteis para os clientes da classe) deverá sempre explicitar informação útil para quem tiver de alterar a classe.

ATENÇÃO: ver na página seguinte como usar o *AgentASD-alunos.zip* que vos é disponibilizado, contendo vários pacotes com ficheiros java.

Como entregar

Fazer **upload** do código Java produzido no **Moodle**, através de um ficheiro *zip* contendo TODAS as classes do projeto. **Este ficheiro zip deverá ter o nome ADSxxx** (onde **xxx** é o número do vosso grupo). Somente um dos elementos do grupo deve fazer o *upload*.

Acrescentem também na classe **startup.Startup** comentários com a indicação do número do vosso grupo e dos números e nomes dos elementos que o compõem.

Quando entregar

O Moodle permitirá o *upload* dos projetos até às 23h55 do dia 28 de Maio.

O que não devem esquecer

A apresentação e documentação da implementação que produzirem e entregarem é **TAMBÉM** importante na medida em que pode facilitar ou dificultar a sua leitura, interpretação e manuseamento. Este aspeto não deve portanto ser descurado!

Qualquer forma de plágio implica a não aprovação na disciplina. Plágio inclui, entre outras formas, copiar ou deixar copiar trabalhos de desenho ou programação. Cada grupo de trabalho é responsável por assegurar que os seus trabalhos são atribuíveis apenas a si.

Como usar as classes fornecidas no *AgentADS-alunos.zip*


- Assegurar-se que tem uma versão do java 17 ou posterior.
- Fazer download do Java FX:
 - No link <https://gluonhq.com/products/javafx/> escolher o adequado ao vosso sistema operativo;
 - Proceder à instalação, escolhendo o local no vosso disco (precisarão de o localizar mais tarde);
- Descomprimir o ficheiro *AgentADS-alunos.zip* fornecido, que contém várias pastas com ficheiros java.

No Eclipse:

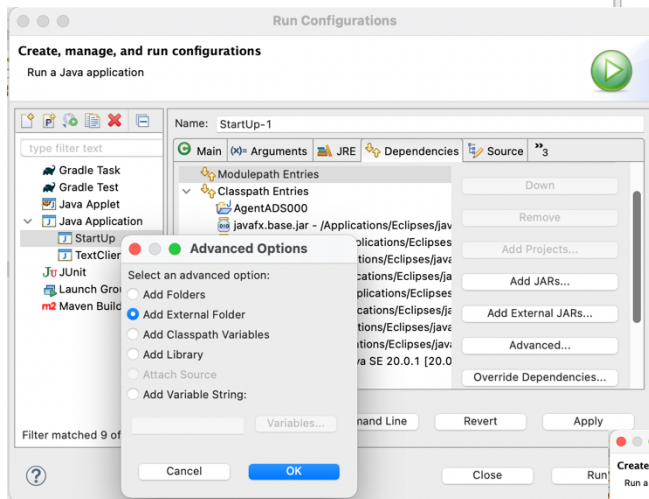
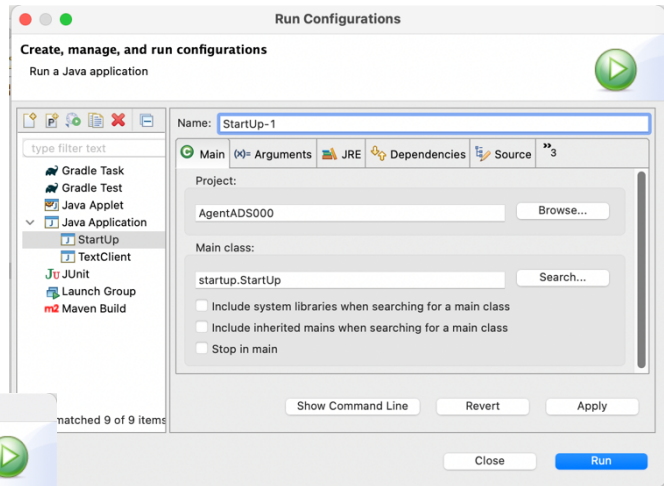
- Criar um projeto Java
 - Dar-lhe o nome ADSxxx (onde xxx é o número do grupo);
 - Não criar module-info na criação do projeto;
 - Certifiquem-se que o JRE é 17 ou superior (o último é 20); (*)
- Copiar as pastas contidas no zip fornecido para a pasta src do vosso novo projeto
 - Pode fazê-lo por arrastamento;
 - São 5 pastas (*controller*, *domain*, *services*, *startup*, *view*), que dão origem a 7 pacotes no projeto java;
 - Os pacotes *controller*, *startup* e *view* apresentam erros de compilação.
- Selecionar o projeto e escolher Properties
 - Selecionar opção “Java Build Path” à esquerda na janela
 - Selecionar a aba “Libraries” em cima
 - Selecionar “ClassPath” na janela, e o botão “Add external jars” à direita
 - Procurar na janela de ficheiros a pasta *lib* do *javafx-sdk-20.0.1*
 - De dentro dessa pasta selecionar todos os ficheiros *.jar* acessíveis e clicar em Open
 - De seguida, Apply and close. Verá os erros de compilação a desaparecer.

(*) Se instalarem a nova versão do java no vosso computador, depois, no Eclipse, escolham essa nova versão em Eclipse → Preferences → Java → Installed JREs

No Eclipse, quando quiser executar a aplicação:

- Se quiser executar o main da classe `startup.TextClient`, pode fazê-lo como normalmente;
- Se quiser executar o main da classe `startup.StartUp`, (o que tem uma interface gráfica) deve:
 - Selecionar a classe `startup.StartUp`;
 - Menu Run → Run configurations;
 - Carregar em  para criar uma nova configuração;

- Dê um nome à nova configuração;
- Na aba “Main”, certifique-se que o projeto e a Main class estão corretas (pode ter que tenha que usar Browse e Search para isso);

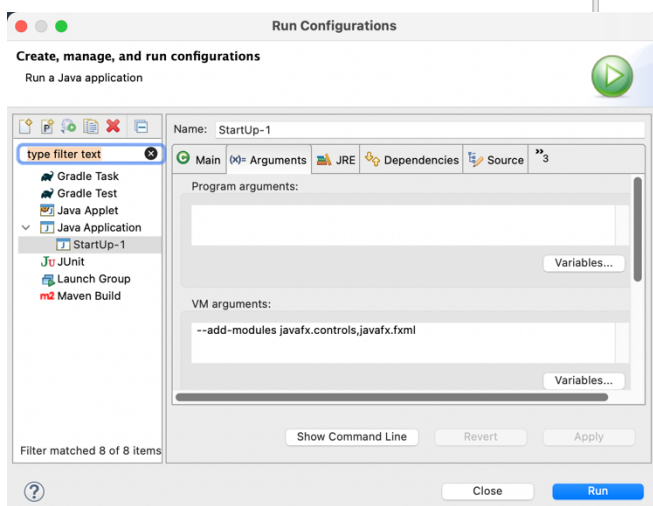
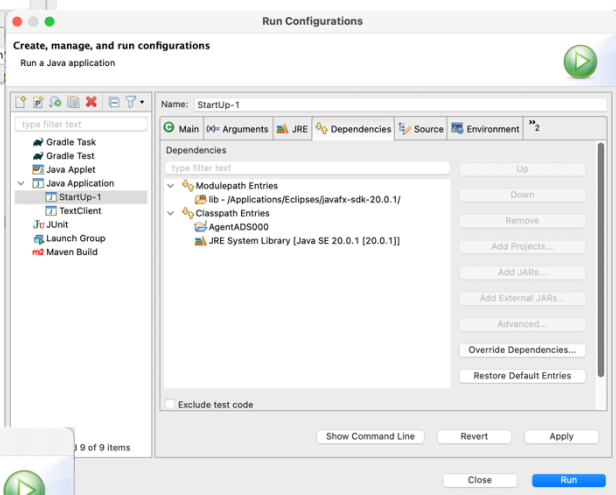


- De seguida, na aba “Dependencies”, carregar no “Modulepath Entries” e escolher o botão “Advanced” à direita;

- Na janela “Advanced Options” escolher “Add External Folder” e, de seguida, na janela do browser, escolher no browser a pasta `lib` do pacote `javafx-sdk-20.0.1` (pode ter que subir um nível para seleccionar a própria pasta `lib`); Clicar em Open;

- Carregar em Apply;

- De seguida, seleccionar todos os `.jar` do `javafx` que estão nas “Classpath Entries” e clicar no botão Remove à direita;
- Carregar em Apply;



- De seguida, na aba “(x)=Arguments”, escrever, na caixa “VM arguments”,

```
--add-modules  
    javafx.controls,javafx.fxml;
```

- Carregar em Apply;

- Finalmente, carregar em Run.

Anexo I: Output do Programa de Teste

A classe `TextClient` do pacote `startup` tem um método `main` que testa a autenticação de utilizadores, a criação de agentes e missões, a consulta a info de agentes e missões, a criação de documentos para agentes, a consulta de documentos e o registo de indisponibilidade de agentes.

Mostra-se de seguida o resultado (a menos de algumas linhas em branco, pois retirámo-las todas aqui para poupar espaço) de uma execução do método `main` desta classe `TextClient`, quando já tiver implementado todas as classes do `domain` e substituído, no método `startup()` desta classe, a instrução `objIni = new domain.stubs.OCT();` por `objIni = new domain.OCT();`.

```
=====
=====      USERS INFO      =====
=====
Name: Peter   Pwd: ABC
Has access to the following agents:
Has access to the following missions:
Has received the following alerts:
-----
Name: Mary    Pwd: 123
Has access to the following agents:
Has access to the following missions:
Has received the following alerts:
-----
Name: John Snow   Pwd: 123ABC
Has access to the following agents:
Has access to the following missions:
Has received the following alerts:
=====
The following password strength verifiers are available
Exigent
Dummy
EasyGoing
Very weak password. Repeat
The following languages are admissible
Portuguese
English
French
Spanish
German
The following codifiers are available
Vigenere
Identity
Round13
The following users exist:
John Snow
Peter
Mary
=====
=====      USERS INFO      =====
=====
Name: Peter   Pwd: ABC
Has access to the following agents:
Name: Houdini   AccessKey: 111
Availability: AVAILABLE   Codifier: Round13   CodingKey: 111
Spoken Languages: French   German
Documents:
From the agent:
To the agent:

Has access to the following missions:
Has received the following alerts:
-----
Name: Mary    Pwd: 123
Has access to the following agents:
Name: Houdini   AccessKey: 111
Availability: AVAILABLE   Codifier: Round13   CodingKey: 111
Spoken Languages: French   German
Documents:
From the agent:
To the agent:

Name: Darth Vader   AccessKey: 999
Availability: AVAILABLE   Codifier: Identity   CodingKey: 666
Spoken Languages: English   Portuguese
Documents:
```


From the agent:
To the agent:

Has access to the following missions:
Has received the following alerts:

Name: John Snow Pwd: 123ABC
Has access to the following agents:
Name: Houdini AccessKey: 111
Availability: AVAILABLE Codifier: Round13 CodingKey: 111
Spoken Languages: French German
Documents:
From the agent:
To the agent:

Name: Darth Vader AccessKey: 999
Availability: AVAILABLE Codifier: Identity CodingKey: 666
Spoken Languages: English Portuguese
Documents:
From the agent:
To the agent:
Has access to the following missions:
Has received the following alerts:

=====

Very weak password. Repeat
The agent does not exist or you do not have access to its info. Repeat

=====

===== USERS INFO =====

=====

Name: Peter Pwd: ABC
Has access to the following agents:
Name: Houdini AccessKey: 111
Availability: AVAILABLE Codifier: Round13 CodingKey: 111
Spoken Languages: French German
Documents:
From the agent:
To the agent:

Has access to the following missions:
Has received the following alerts:

Name: Mary Pwd: 123
Has access to the following agents:
Name: Houdini AccessKey: 111
Availability: AVAILABLE Codifier: Round13 CodingKey: 111
Spoken Languages: French German
Documents:
From the agent:
To the agent:

Name: Darth Vader AccessKey: 999
Availability: AVAILABLE Codifier: Identity CodingKey: 666
Spoken Languages: English Portuguese
Documents:
From the agent:
To the agent:

Has access to the following missions:
Name: Roling Stones AccessKey: rolSto
Responsible agent: Houdini KeyWords:
Africa Trafico
Participating agents:
Name: Darth Vader AccessKey: 999
Availability: AVAILABLE Codifier: Identity CodingKey: 666
Spoken Languages: English Portuguese
Documents:
From the agent:
To the agent:

Has received the following alerts:

Name: John Snow Pwd: 123ABC
Has access to the following agents:
Name: Houdini AccessKey: 111
Availability: AVAILABLE Codifier: Round13 CodingKey: 111
Spoken Languages: French German
Documents:
From the agent:
To the agent:

Name: Darth Vader AccessKey: 999
Availability: AVAILABLE Codifier: Identity CodingKey: 666
Spoken Languages: English Portuguese
Documents:
From the agent:
To the agent:

Has access to the following missions:
Name: Roling Stones AccessKey: rolSto
Responsible agent: Houdini Keywords:
Africa Trafico
Participating agents:
Name: Darth Vader AccessKey: 999
Availability: AVAILABLE Codifier: Identity CodingKey: 666
Spoken Languages: English Portuguese
Documents:
From the agent:
To the agent:

Has received the following alerts:
=====

Codifier: Identity
Spoken languages: English Portuguese
=====

===== USERS INFO =====
=====

Name: Peter Pwd: ABC
Has access to the following agents:
Name: Houdini AccessKey: 111
Availability: AVAILABLE Codifier: Round13 CodingKey: 111
Spoken Languages: French German
Documents:
From the agent:
To the agent:

Has access to the following missions:
Has received the following alerts:

Name: Mary Pwd: 123
Has access to the following agents:
Name: Houdini AccessKey: 111
Availability: AVAILABLE Codifier: Round13 CodingKey: 111
Spoken Languages: French German
Documents:
From the agent:
To the agent:

Name: Darth Vader AccessKey: 999
Availability: AVAILABLE Codifier: Identity CodingKey: 666
Spoken Languages: English Portuguese
Documents:
From the agent:
To the agent:

Has access to the following missions:
Name: Roling Stones AccessKey: rolSto
Responsible agent: Houdini Keywords:
Africa Trafico
Participating agents:
Name: Darth Vader AccessKey: 999
Availability: AVAILABLE Codifier: Identity CodingKey: 666
Spoken Languages: English Portuguese
Documents:
From the agent:
To the agent:

Has received the following alerts:
Agent Darth Vader codifier name was searched

Name: John Snow Pwd: 123ABC
Has access to the following agents:
Name: Houdini AccessKey: 111
Availability: AVAILABLE Codifier: Round13 CodingKey: 111
Spoken Languages: French German
Documents:
From the agent:
To the agent:

Name: Darth Vader AccessKey: 999
Availability: AVAILABLE Codifier: Identity CodingKey: 666

Spoken Languages: English Portuguese
Documents:
From the agent:
To the agent:

Has access to the following missions:
Name: Roling Stones AccessKey: rolSto
Responsible agent: Houdini Keywords:
Africa Trafico
Participating agents:
Name: Darth Vader AccessKey: 999
Availability: AVAILABLE Codifier: Identity CodingKey: 666
Spoken Languages: English Portuguese
Documents:
From the agent:
To the agent:

Has received the following alerts:
Agent Darth Vader codifier name was searched
=====

Responsible agent: Houdini
Participating agents: Darth Vader
Mission characteristics: Africa Trafico
=====

===== USERS INFO =====
=====

Name: Peter Pwd: ABC
Has access to the following agents:
Name: Houdini AccessKey: 111
Availability: AVAILABLE Codifier: Round13 CodingKey: 111
Spoken Languages: French German
Documents:
From the agent:
To the agent:

Has access to the following missions:
Has received the following alerts:

Name: Mary Pwd: 123
Has access to the following agents:
Name: Houdini AccessKey: 111
Availability: AVAILABLE Codifier: Round13 CodingKey: 111
Spoken Languages: French German
Documents:
From the agent:
To the agent:

Name: Darth Vader AccessKey: 999
Availability: AVAILABLE Codifier: Identity CodingKey: 666
Spoken Languages: English Portuguese
Documents:
From the agent:
To the agent:

Has access to the following missions:
Name: Roling Stones AccessKey: rolSto
Responsible agent: Houdini Keywords:
Africa Trafico
Participating agents:
Name: Darth Vader AccessKey: 999
Availability: AVAILABLE Codifier: Identity CodingKey: 666
Spoken Languages: English Portuguese
Documents:
From the agent:
To the agent:

Has received the following alerts:
Agent Darth Vader codifier name was searched
Mission Roling Stones info was searched

Name: John Snow Pwd: 123ABC
Has access to the following agents:
Name: Houdini AccessKey: 111
Availability: AVAILABLE Codifier: Round13 CodingKey: 111
Spoken Languages: French German
Documents:
From the agent:
To the agent:

Name: Darth Vader AccessKey: 999

Availability: AVAILABLE Codifier: Identity CodingKey: 666
 Spoken Languages: English Portuguese
 Documents:
 From the agent:
 To the agent:
 Has access to the following missions:
 Name: Roling Stones AccessKey: rolSto
 Responsible agent: Houdini KeyWords:
 Africa Trafico
 Participating agents:
 Name: Darth Vader AccessKey: 999
 Availability: AVAILABLE Codifier: Identity CodingKey: 666
 Spoken Languages: English Portuguese
 Documents:
 From the agent:
 To the agent:

Has received the following alerts:
 Agent Darth Vader codifier name was searched
 Mission Roling Stones info was searched
 =====
 The agent is presently AVAILABLE
 Kinds of availability: AVAILABLE ARRESTED DESERTED ILL DEAD RETIRED
 =====
 ===== USERS INFO =====
 =====

Name: Peter Pwd: ABC
 Has access to the following agents:
 Name: Houdini AccessKey: 111
 Availability: AVAILABLE Codifier: Round13 CodingKey: 111
 Spoken Languages: French German
 Documents:
 From the agent:
 To the agent:

Has access to the following missions:
 Has received the following alerts:

Name: Mary Pwd: 123
 Has access to the following agents:
 Name: Houdini AccessKey: 111
 Availability: AVAILABLE Codifier: Round13 CodingKey: 111
 Spoken Languages: French German
 Documents:
 From the agent:
 To the agent:

Name: Darth Vader AccessKey: 999
 Availability: ARRESTED Codifier: Identity CodingKey: 666
 Spoken Languages: English Portuguese
 Documents:
 From the agent:
 To the agent:

Has access to the following missions:
 Name: Roling Stones AccessKey: rolSto
 Responsible agent: Houdini KeyWords:
 Africa Trafico
 Participating agents:
 Has received the following alerts:
 Agent Darth Vader codifier name was searched
 Mission Roling Stones info was searched
 Agent Darth Vader became ARRESTED

Name: John Snow Pwd: 123ABC
 Has access to the following agents:
 Name: Houdini AccessKey: 111
 Availability: AVAILABLE Codifier: Round13 CodingKey: 111
 Spoken Languages: French German
 Documents:
 From the agent:
 To the agent:

Name: Darth Vader AccessKey: 999
 Availability: ARRESTED Codifier: Identity CodingKey: 666
 Spoken Languages: English Portuguese
 Documents:
 From the agent:
 To the agent:

Has access to the following missions:
Name: Roling Stones AccessKey: rolSto
Responsible agent: Houdini KeyWords:
Africa Trafico
Participating agents:
Has received the following alerts:
Agent Darth Vader codifier name was searched
Mission Roling Stones info was searched
Agent Darth Vader became ARRESTED
=====

The reference of the new document is: ToAg0
=====

===== USERS INFO =====
=====

Name: Peter Pwd: ABC
Has access to the following agents:
Name: Houdini AccessKey: 111
Availability: AVAILABLE Codifier: Round13 CodingKey: 111
Spoken Languages: French German
Documents:
From the agent:
To the agent:
Reference: ToAg0 Codified text:
Ba gur pbeare bs gur lfg jvgu gur 3eq
ng sbhe bs gur nsgreabba
jrnevat n erq pbng naq n oynpx ung

Has access to the following missions:
Has received the following alerts:

Name: Mary Pwd: 123
Has access to the following agents:
Name: Houdini AccessKey: 111
Availability: AVAILABLE Codifier: Round13 CodingKey: 111
Spoken Languages: French German
Documents:
From the agent:
To the agent:
Reference: ToAg0 Codified text:
Ba gur pbeare bs gur lfg jvgu gur 3eq
ng sbhe bs gur nsgreabba
jrnevat n erq pbng naq n oynpx ung

Name: Darth Vader AccessKey: 999
Availability: ARRESTED Codifier: Identity CodingKey: 666
Spoken Languages: English Portuguese
Documents:
From the agent:
To the agent:

Has access to the following missions:
Name: Roling Stones AccessKey: rolSto
Responsible agent: Houdini KeyWords:
Africa Trafico
Participating agents:

Has received the following alerts:
Agent Darth Vader codifier name was searched
Mission Roling Stones info was searched
Agent Darth Vader became ARRESTED

Name: John Snow Pwd: 123ABC
Has access to the following agents:
Name: Houdini AccessKey: 111
Availability: AVAILABLE Codifier: Round13 CodingKey: 111
Spoken Languages: French German
Documents:
From the agent:
To the agent:
Reference: ToAg0 Codified text:
Ba gur pbeare bs gur lfg jvgu gur 3eq
ng sbhe bs gur nsgreabba
jrnevat n erq pbng naq n oynpx ung

Name: Darth Vader AccessKey: 999
Availability: ARRESTED Codifier: Identity CodingKey: 666
Spoken Languages: English Portuguese
Documents:
From the agent:
To the agent:

```

Has access to the following missions:
Name: Roling Stones   AccessKey: rolSto
Responsible agent: Houdini   Keywords:
Africa   Trafico
Participating agents:
Has received the following alerts:
Agent Darth Vader codifier name was searched
Mission Roling Stones info was searched
Agent Darth Vader became ARRESTED
=====
References of available docs: ToAg0
The selected document, already decodified:
On the corner of the 1st with the 3rd
at four of the afternoon
wearing a red coat and a black hat
=====
=====      USERS INFO      =====
=====
Name: Peter   Pwd: ABC
Has access to the following agents:
Name: Houdini   AccessKey: 111
Availability: AVAILABLE   Codifier: Round13   CodingKey: 111
Spoken Languages: French   German
Documents:
From the agent:
To the agent:
Reference: ToAg0   Codified text:
Ba gur pbeare bs gur lfg jvgu gur 3eq
ng sbhe bs gur nsgreabba
jrnevat n erq pbng naq n oynpx ung

Has access to the following missions:
Has received the following alerts:
Agent Houdini doc references were searched
Agent Houdini document ToAg0 was searched
-----
Name: Mary   Pwd: 123
Has access to the following agents:
Name: Houdini   AccessKey: 111
Availability: AVAILABLE   Codifier: Round13   CodingKey: 111
Spoken Languages: French   German
Documents:
From the agent:
To the agent:
Reference: ToAg0   Codified text:
Ba gur pbeare bs gur lfg jvgu gur 3eq
ng sbhe bs gur nsgreabba
jrnevat n erq pbng naq n oynpx ung

Name: Darth Vader   AccessKey: 999
Availability: ARRESTED   Codifier: Identity   CodingKey: 666
Spoken Languages: English   Portuguese
Documents:
From the agent:
To the agent:

Has access to the following missions:
Name: Roling Stones   AccessKey: rolSto
Responsible agent: Houdini   Keywords:
Africa   Trafico
Participating agents:

Has received the following alerts:
Agent Darth Vader codifier name was searched
Mission Roling Stones info was searched
Agent Darth Vader became ARRESTED
Agent Houdini doc references were searched
Agent Houdini document ToAg0 was searched
-----
Name: John Snow   Pwd: 123ABC
Has access to the following agents:
Name: Houdini   AccessKey: 111
Availability: AVAILABLE   Codifier: Round13   CodingKey: 111
Spoken Languages: French   German
Documents:
From the agent:
To the agent:
Reference: ToAg0   Codified text:
Ba gur pbeare bs gur lfg jvgu gur 3eq

```

ng sbhe bs gur nsGREAbba
jrnevAt n erq pbng naq n oYnpX ung

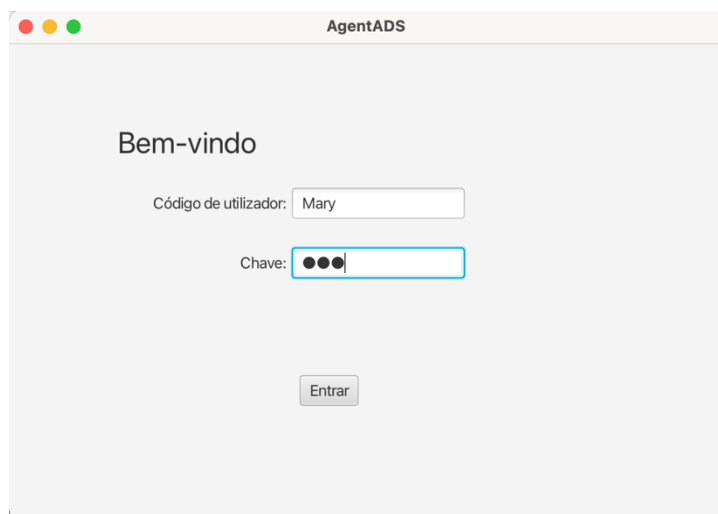
Name: Darth Vader AccessKey: 999
Availability: ARRESTED Codifier: Identity CodingKey: 666
Spoken Languages: English Portuguese
Documents:
From the agent:
To the agent:

Has access to the following missions:
Name: Roling Stones AccessKey: rolSto
Responsible agent: Houdini KeyWords:
Africa Trafico
Participating agents:

Has received the following alerts:
Agent Darth Vader codifier name was searched
Mission Roling Stones info was searched
Agent Darth Vader became ARRESTED
Agent Houdini doc references were searched
Agent Houdini document ToAg0 was searched
=====

Anexo II – Breve explicação da interface de utilizador

A interface de utilizador desenvolvida é muito simples e padrão, pelo que não se espera que seja difícil de utilizar. Ao arrancar com a aplicação (executando a classe `startup.Startup`) aparece o seguinte ecrã:



Este ecrã implementa o caso de uso *Login de utilizador*.

Na versão que vos fornecemos é possível autenticar-se com qualquer *username* e palavra-passe, exceto com o *username* “fail”, que causa o aparecimento de uma janela de erro a simular o que ocorre quando o utilizador se engana no *username* ou palavra-passe.

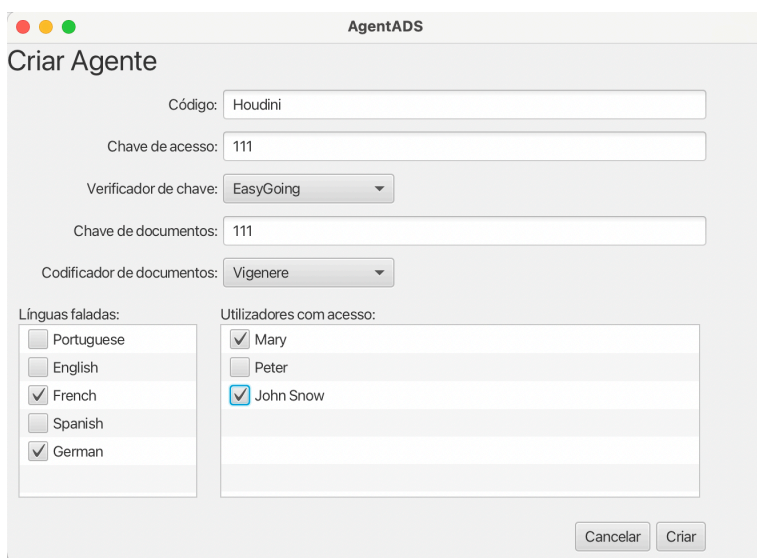
Claro que quando for usada a implementação correta do objeto inicial `domain.OCT`, em vez de `domain.stubs.OCT`, já só serão aceites os users Mary, Peter e John Snow...

Pode agora seleccionar no menu “Menu” a opção “Criar Agente”. Esta opção permite executar o caso de uso *Criar novo Agente* e o ecrã tem o seguinte aspeto.

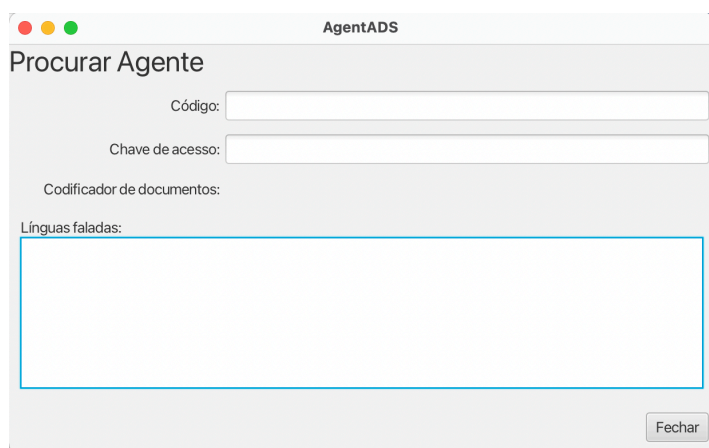
Há campos que dependem do preenchimento de outros, campos que impedem o preenchimento de outros, etc.

Experimentem alterar valores já recolhidos anteriormente e notem o que aparece escrito na consola sobre a execução das operações do caso de uso.

Como a interface gráfica é muito liberal e permite alterar qualquer campo a qualquer altura, quando a alteração não é feita pela sequência do caso de uso, este é cancelado e uma nova execução do caso de uso é iniciada, mas não têm de introduzir a informação de novo.



A execução do caso de uso *Consultar Info de Agente* é conseguida a partir da seleção da opção “Procurar informação de agente” do menu da aplicação. O ecrã que aparece tem o seguinte aspeto:

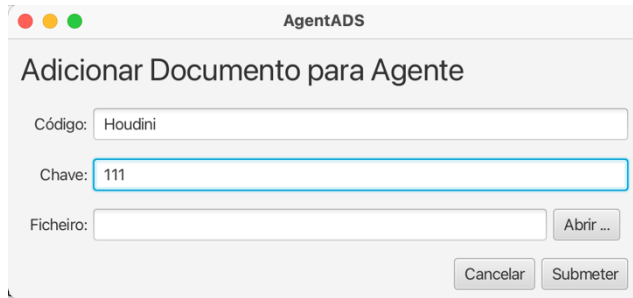


Aqui deve-se introduzir o nome de código e a chave de acesso de um agente.

Quando se carrega em *enter*, aparece o nome do codificador desse agente e as línguas por ele faladas.

A aplicação está feita de modo a avisar se alguma das seguintes situações ocorrer: não existe nenhum agente com esse nome de código ou existe um mas a chave de acesso não é a correta ou, ainda, o utilizador autenticado não tem permissão de acesso a esse agente.

Para executar o caso de uso *Adicionar Documento para Agente* deve-se selecionar a opção “Adicionar documento para agente” do menu da aplicação. O ecrã que aparece tem o seguinte aspeto:



De novo, aqui deve-se introduzir o nome de código e a chave de acesso de um agente.

De seguida deve-se indicar o nome do ficheiro contendo o texto do documento que se quer adicionar.

Isso pode ser feito usando o botão “Abrir” que dá acesso a um browser para seleção de ficheiros.

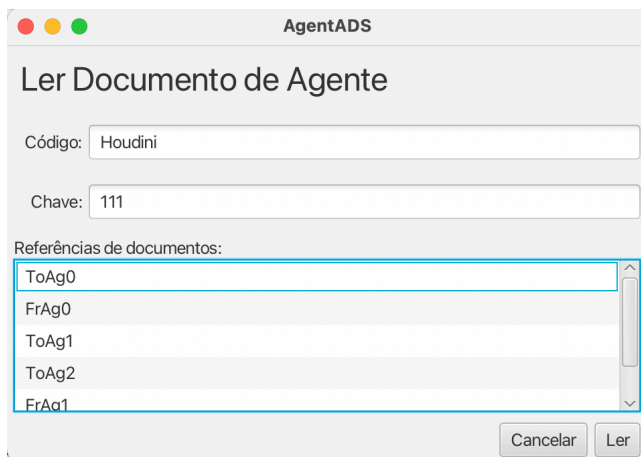
O botão “Submeter” permite registar um

novo documento, já codificado, para esse agente.

É mostrado o seguinte ecrã, indicando a referência que foi atribuída ao novo documento:



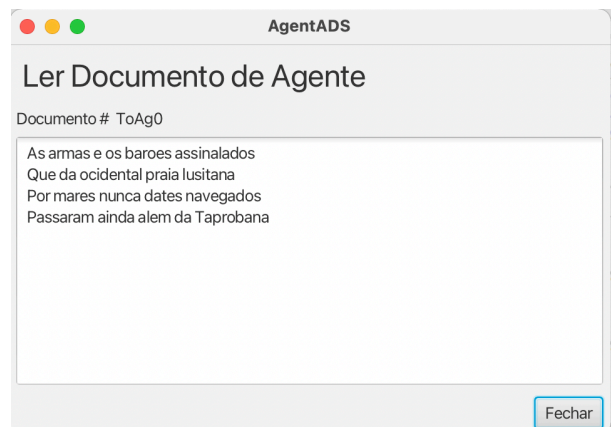
Para executar o caso de uso *Ler Documento de Agente* deve-se selecionar a opção “Ler documento de agente” do menu da aplicação. O ecrã que aparece tem o seguinte aspeto:



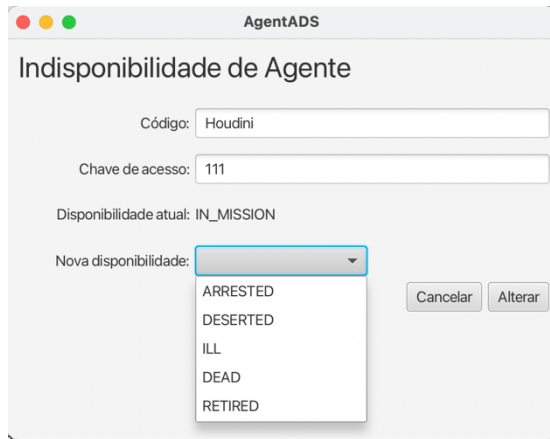
De novo, aqui deve-se introduzir o nome de código e a chave de acesso de um agente e carregar em enter.

De seguida deve-se escolher uma das referências a documentos que aparecem listadas e carregar em “Ler”.

O seguinte ecrã aparece, com o texto, já decodificado, do documento escolhido:



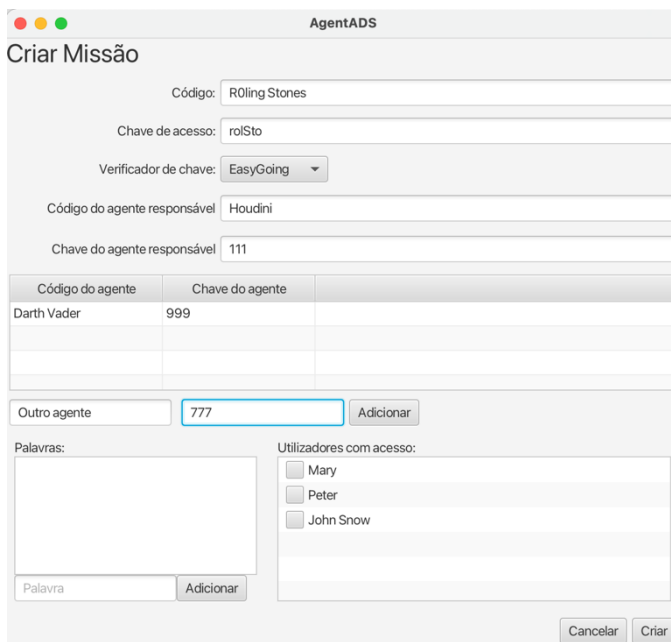
Para executar o caso de uso *Registar Indisponibilidade de Agente* deve-se seleccionar a opção “Registar indisponibilidade de agente” do menu da aplicação. O ecrã que aparece tem o seguinte aspeto:



De novo, aqui deve-se introduzir o nome de código e a chave de acesso de um agente e carregar em *enter*.

A aplicação mostra a disponibilidade atual do agente e de seguida o utilizador deve escolher uma das indisponibilidades possíveis e carregar em “Alterar”.

Para executar o caso de uso *Criar nova Missão* deve-se seleccionar a opção “Criar missão” do menu da aplicação. O ecrã que aparece tem o seguinte aspeto:



O utilizador deve introduzir o nome de código e a chave de acesso para a nova missão, bem como escolher o verificador de “força” da chave de acesso.

Deve ainda indicar o nome de código e a chave de acesso do agente que vai ser o responsável pela missão.

De seguida deve escrever nomes de código e chaves de acesso dos agentes participantes, um de cada vez, e carregar em “Adicionar”, o que faz com que sejam, um a um, adicionados à lista dos agentes.

De seguida deve fazer o mesmo para as palavras ou características que definem a missão e por fim deve escolher quais os utilizadores que poderão ter acesso à informação desta missão. No fim deve carregar em “Criar”.

Por fim, para executar o caso de uso *Consultar Info de Missão* deve-se seleccionar a opção “Procurar informação de missão” do menu da aplicação.

No ecrã que aparece deve-se preencher o nome de código e a chave de acesso da missão e carregar em *enter*.

Será então apresentada a informação sobre o agente responsável, os agentes participantes e as características da missão.

