|  |  |  |
| --- | --- | --- |
| **Módulo:** | Inteligência artificial | |
| **Objectivo:** | Testar listagem de amigos de um utilizador | |
| **Métodos de teste:** | Manual | |
| **Cenário** | **Teste** | **Resultado esperado** |
| Base de conhecimento “lapr.pl” | listar\_am(a). | b,c,d,e |
| Listar\_am(z). | --- |
| Base de conhecimento “recomendaAmigos.pl” | Listar\_am(10). | 1 |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Módulo:** | Inteligência artificial | | |
| **Objectivo:** | Testar listagem de amigos sugeridos a um utilizador (com *tags* em comum entre os dois utilizadores) | | |
| **Métodos de teste:** | Automático | | |
| **Cenário** | **Teste** | **Resultado esperado** | **Resultado**  **obtido** |
| Base de conhecimento “recomendaAmigos.pl” | sugerir\_users(10). | 7[rock],20[pimba] | 7[rock],20[pimba] |
| sugerir\_users(7). | 20[terror] | 20[terror] |
| sugerir\_users(100). | --- | --- |

|  |  |  |  |
| --- | --- | --- | --- |
| **Módulo:** | Inteligência artificial | | |
| **Objectivo:** | Determinar o menor caminho entre dois utilizadores | | |
| **Métodos de teste:** | Automático | | |
| **Cenário** | **Teste** | **Resultado esperado** | **Resultado**  **obtido** |
| Base de conhecimento “menorCaminho.pl” | cam\_min(10,7,L). | L=[10,7] | L=[10,7] |
| cam\_min (10,20,L). | L=[10,7,20] | L=[10,7,20] |
| cam\_min (10,100,L). | --- | --- |
| cam\_min (10,21,L). | L=[10,7,21] | L=[10,7,21] |

|  |  |  |  |
| --- | --- | --- | --- |
| **Módulo:** | Inteligência artificial | | |
| **Objectivo:** | Determinar o menor caminho entre dois utilizadores | | |
| **Métodos de teste:** | Automático | | |
| **Cenário** | **Teste** | **Resultado esperado** | **Resultado**  **obtido** |
| Base de conhecimento “tamanhoRedeUtilizador.pl” | tamanho\_rede\_user\_3(10,N). | N=4 | N=4 |
| tamanho\_rede\_user\_3(4,N). | *no* | *no* |
| tamanho\_rede\_user\_3(20,N). | N=3 | N=3 |
| tamanho\_rede\_user\_3(11,A). | A=2 | A=2 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Módulo:** | Inteligência artificial | | |
| **Objectivo:** | Listar os amigos até terceiro nível de um utilizador (imprime lista) | | |
| **Métodos de teste:** | Manual | | |
| **Cenário** | **Teste** | **Resultado esperado** | **Resultado**  **obtido** |
| Base de conhecimento “lapr.pl” | rede\_user\_3 (a,N). | [b,c,d,e] | [b,c,d,e] |
| rede\_user\_3 (b,N). | [c,d,e,f] | [c,d,e,f] |
| rede\_user\_3 (z,N). | *no* | *no* |

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

|  |  |  |  |
| --- | --- | --- | --- |
| **Módulo:** | Inteligência artificial | | |
| **Objectivo:** | Listar os amigos até terceiro nível de um utilizador (imprime lista) | | |
| **Métodos de teste:** | Manual | | |
| **Cenário** | **Teste** | **Resultado esperado** | **Resultado**  **obtido** |
| Base de conhecimento “caminho mais forte.pl” | camMaisForte(a,d,L). | L=[a,c,d] | L=[a,c,d] |
| camMaisForte(a,ww,L). | Nenhum caminho | Nenhum caminho |
| camMaisForte(a,a,L). | Nenhum caminho | Nenhum caminho |

|  |  |  |  |
| --- | --- | --- | --- |
| **Módulo:** | Inteligência artificial | | |
| **Objectivo:** | Obter o utilizador estrela de uma tag | | |
| **Métodos de teste:** | Manual | | |
| **Cenário** | **Teste** | **Resultado esperado** | **Resultado**  **obtido** |
| Base de conhecimento “maven.pl” | maven('rock',U). | U = e | U = e |
| maven('roc',U). | Nenhum Maven | Nenhum Maven... |
| maven('pimba',U). | Nenhum Maven | Nenhum Maven... |

|  |  |  |  |
| --- | --- | --- | --- |
| **Módulo:** | Inteligência artificial / C # | | |
| **Objectivo:** | Obter os nós para o ficheiro pl | | |
| **Métodos de teste:** | Manual | | |
| **Cenário** | **Teste** | **Resultado esperado** | **Resultado**  **obtido** |
| Simular o webservice a ir buscar os nós a base de dados para gerar o ficheiro pl | obterNosPublic() | Nós da base de dados  no(noID,[Tags],X,Y). | no(noID,[Tags],X,Y)./n |
|  |  |  |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Módulo:** | Inteligência artificial / C # | | |
| **Objectivo:** | Obter os ramos para o ficheiro pl | | |
| **Métodos de teste:** | Manual | | |
| **Cenário** | **Teste** | **Resultado esperado** | **Resultado**  **obtido** |
| Simular o webservice a ir buscar os ramos a base de dados para gerar o ficheiro pl | obterRamosPublic() | Nós da base de dados  Ramo  (IDA,IDB,[Tags],força) | Ramo  (IDA,IDB,[Tags],força)./n |
|  |  |  |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Módulo:** | Inteligência artificial / C # | | |
| **Objectivo:** | Obter o conteúdo do ficheiro prolog conforme o pedido | | |
| **Métodos de teste:** | Automático | | |
| **Cenário** | **Teste** | **Resultado esperado** | **Resultado**  **obtido** |
| Simular o webservice a ir buscar o ficheiro de texto para gerar o ficheiro pl.  2º teste erro na ligação ao base de dados  3º não encontra prolog pedido | obterLines(target) | Nos  Ramos  Metodo Prolog  Run | Nos  Ramos  Metodo Prolog  Run |
| obterLines(target) | Null | Null |
| obterLines(target) | Null | Null |

|  |  |  |  |
| --- | --- | --- | --- |
| **Módulo:** | Inteligência artificial / C # | | |
| **Objectivo:** | Executar um comando prolog autonomamente. | | |
| **Métodos de teste:** | Automático | | |
| **Cenário** | **Teste** | **Resultado esperado** | **Resultado**  **obtido** |
| Simular o webservice a executar um comando prolog, fornecendo o id do utilizador que faz o pedido e o nome do comando (ie. menorCaminho, grafoUser) | PrologExec p = new PrologExec("user", "maven");  p.executaComandoProlog("musica"); | 7 | 7 |
| PrologExec p = new PrologExec("user", "maven");  p.executaComandoProlog("rock"); | 1 | 1 |
| PrologExec p = new PrologExec("user", "maven");  p.executaComandoProlog("m"); | Nenhum maven | Nenhum maven |

|  |  |  |  |
| --- | --- | --- | --- |
| **Módulo:** | Webservice / C++ | | |
| **Objectivo:** | Retornar o profileid do utilizador que faz login | | |
| **Métodos de teste:** | Automático | | |
| **Cenário** | **Teste** | **Resultado esperado** | **Resultado**  **obtido** |
| Webservice vai ao membership do site para validar o username | validateLogin  ("RafaUser",”qwerty”) | 1 | 1 |
| validateLogin  ("RafaUser",qwert) | 0 | 0 |
| validateLogin  ("",””) | 0 | 0 |