Upskill - Java

Laboratório de Programação

T4J – TASKS FOR JOE

Novos Requisitos

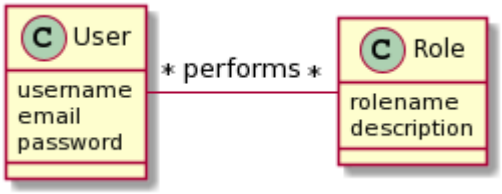
Durante o período de apresentação de candidaturas previsto em cada anúncio, para além de efetuar candidaturas, os freelancers podem:

- Atualizar a informação das candidaturas previamente efetuadas por si;
- Retirar candidaturas previamente efetuadas por si;

GESTÃO DE UTILIZADORES

Pretende-se que seja desenvolvida uma REST API para a gestão de utilizadores. O componente de gestão de utilizadores visa satisfazer um conjunto de requisitos/funcionalidades genéricas e comuns a várias aplicações. A autenticação dos utilizadores será realizada com um identificador de utilizador e palavra-passe.

Os conceitos principais abordados neste componente são apresentados de seguida.



De acordo com o modelo do domínio, um *"User"* desempenha vários *"Role"* (i.e. funções), sendo que o mesmo *"Role"* também pode ser desempenhado por vários *"User"*.

Este componente deverá disponibilizar as suas funcionalidades através de um *Web Service* que terá de estar em conformidade com o estilo arquitetural *REST* (*Representational State Transfer*). A utilização do serviço apenas deverá ser possível a quem conhecer uma chave que é obtida após registo.

Devem ser desenvolvidos os seguintes métodos, *endpoints*, recursos, parâmetros e resultados de output esperados:

Resource	*Method*	*Parameters*	*Output example (Json format)*
/context	GET	app_key : String	[{"app_context":"{7E19F342-A903-4C3B-806A-CF771120B9D0}","timeout": "300"}]
/registerUser	POST	app_context : String username : String email : String password : String	
/registerUserWithRoles	POST	app_context : String username : String email : String password : String rolenames : String	
/login	POST	app_context : String user_id : String password : String	
/logout	POST	app_context : String	
/userRoles	GET	app_context : String user_id : String	{"rolename": "gestor","description": "Gestor de Organização"}]
/userRoles	POST	app_context : String user_id : String rolenames : String	
/userRoles	DELETE	app_context : String user_id : String rolenames : String	
/roles	GET	app_context : String	[{"rolename": "administrativo","description": "Administrativo"}, {"rolename": "colaborador","description": "Colaborador de Organização"}, {"rolename": "freelancer","description": "Freelancer"}, {"rolename": "gestor","description": "Gestor de Organização"}]
/roles	POST	app_context : String rolename : String description : String	
/roles	DELETE	app_context : String rolename : String	
/session	GET	app_context : String	[{"username": "John Doe","email": "jdoe@company.com" ,"rolenames": null, "logindate": "2021-01-23T18:33:27.000Z"}]

O *Web Service* disponibiliza os seguintes recursos:

- ***"GET /context(String app_key)"***: permite obter uma chave de contexto que identifica uma sessão de utilizador. Esta chave de contexto será necessária para executar todos os restantes métodos disponíveis. Esta chave é válida por um período de tempo (em milissegundos|) que é devolvida pela API depois da obtenção do contexto, até que seja efetuado um login ou um registo de utilizador. Caso não seja efetuado o login ou o registo de utilizador no período de tempo especificado, é necessário obter uma nova chave de contexto.
- ***"POST /registerUser(String app_context, String username, String email, String password)"***: permite definir a existência de um novo utilizador que pode usar a aplicação;

- ***"POST /registerUserWithRole(String app_context, String username, String email, String password, String rolenames)"***: semelhante ao método anterior, com a vantagem de associar imediatamente um papel/função a esse utilizador;
- ***"POST login(String app_context, String user_id, String password)"***: permite autenticar um utilizador com vista à utilização da aplicação. O parâmetro *user_id* pode ser o *username* ou o *email* do utilizador;
- ***"POST logout(String app_context)"***: termina a sessão de utilizador que estiver ativa no momento;
- ***"GET /userRoles(String app_context, String user_id)"***: permite obter a lista de papéis/funções associados a um dado utilizador;
- ***"POST /userRoles(String app_context, String user_id, String rolenames)"***: permite associar papeis/funções a um dado utilizador;

  "DELETE /userRoles(String app_context, String user_id, String rolenames)": permite eliminar a associação de papeis/funções a um dado utilizador;
- ***"GET /roles(String app_context)"***: permite obter todos os papéis/funções existentes no sistema;
- ***"POST /roles(String app_context, String rolename, String description)"***: permite definir a existência de um novo papel/função de interesse para a aplicação;
- ***"DELETE /roles(String app_context, String rolename)"***: permite eliminar o papel/função existente no sistema apenas caso este não tenha utilizadores associados;
- ***"GET /session(String app_context)"***: permite obter os dados da sessão ativa.

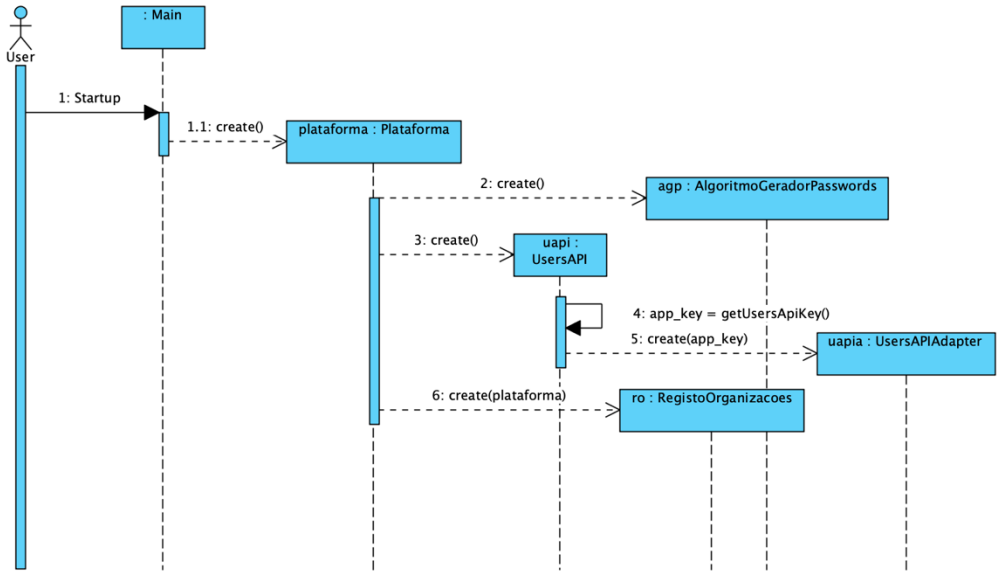
O *Web Service* deverá associar à resposta a todos os pedidos realizados, um *HTTP response status code* em concordância com a publicação RFC 2616, de junho de 1999, que definiu o HTTP/1.1. Por exemplo, devem ser devolvidos os códigos HTTP 200 (OK) a cada pedido corretamente processado, 401 (*unauthorized*) a cada pedido sem autorização (i.e. contexto inválido, credenciais inválidas, etc.) e 500 (*internal* error) para todas as situações representativas de erros internos do serviço.

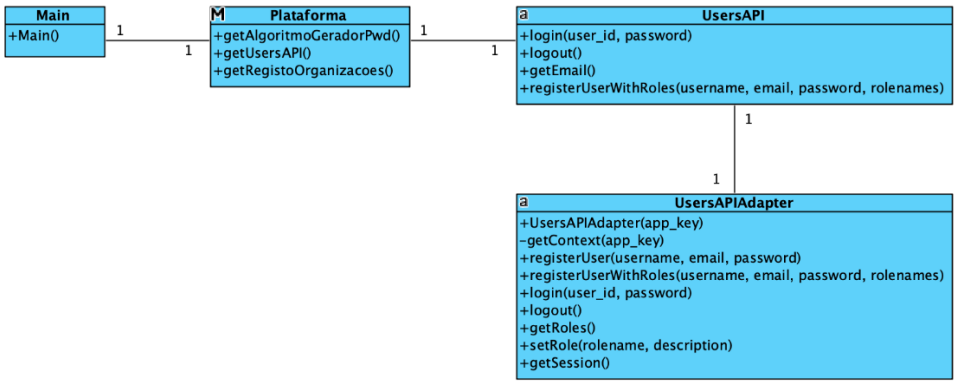
API PARA GESTÃO DE UTILIZADORES

Apresenta-se de seguida uma sugestão de design para a utilização do serviço de gestão de utilizadores.

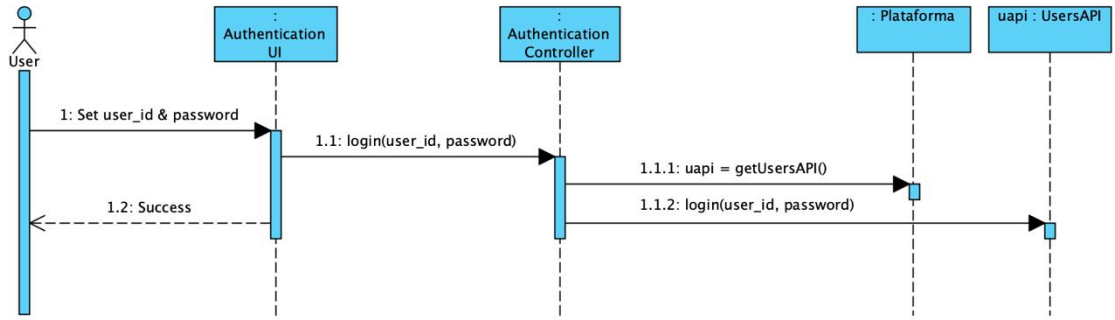
Arranque da aplicação

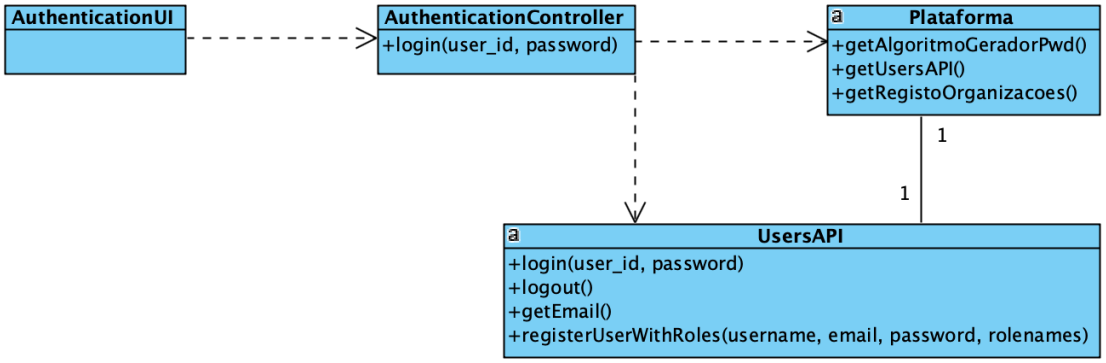
No arranque da aplicação devem ser criados os objetos necessários para o consumo do *webservice*.





Consumo do *webservice* (exemplo: autenticação)





ANÁLISE, DESIGN E IMPLEMENTAÇÃO

Desenvolva uma aplicação que disponibilize as funcionalidades expressas no enunciado acima.

Todo o código produzido deve ter sempre em consideração os principais princípios da programação orientada por objetos: abstração, encapsulamento, herança e polimorfismo.

O núcleo principal do software deve ser implementado em Java. Com o intuito de aumentar a manutenibilidade do software, devem ser adotadas boas práticas de análise e design de software OO.

Aplique o processo de desenvolvimento de software designado por Test Driven Development (TDD) na implementação das classes.

Dever-se-á utilizar o *plugin* Maven JaCoCo (Java Code Coverage) no IDE Netbeans para verificar a cobertura de testes.

O código deverá conter os comentários necessários para que possa ser gerada a documentação usando a ferramenta Javadoc.

A implementação do software deve adotar normas de codificação (e.g. Camel case) e de controlo de versões. O controlo de versões será conseguido, usando o GitHub.

O trabalho deverá ser realizado por um grupo de quatro formandos.

Deverá ser submetido no Moodle do UPskill, um ficheiro ZIP com: o projeto Maven com o seguinte formato:

JavaNºTurma_PrimeiroUltimoNome_PrimeiroUltimoNome_PrimeiroUltimoNome_Primeiro UltimoNome, como por exemplo, Java1_IsabelBras_ArturSilva_CarlaCosta_António_Sousa; e um pdf com a parte restante do design.

O projeto Maven deve ser implementado recorrendo a um repositório do GitHub, criado e configurado por um dos elementos do grupo. Na turma Java1, os professores Jorge Santos (ajs@upskill.pt) , Alexandre Gouveia (aas@upskill.pt), Nuno Morgado (nvm@upskill.pt), Marílio Cardoso (joc@upskill.pt) e Paulo Baltarejo (pbs@upskill.pt) têm de ser adicionados à lista de elementos com acesso ao repositório. Na turma Java2, os professores Nuno Melo e Castro (anc@upskill.pt), Jorge Duarte (fjd@upskill.pt), Alexandre Gouveia (aas@upskill.pt), Nuno Morgado (nvm@upskill.pt), Isabel Sampaio (ais@upskill.pt) e Joaquim Santos (jpe@upskill.pt) têm de ser adicionados à lista de elementos com acesso ao repositório.

Esse ficheiro ZIP deverá conter também mais dois pdf's: um pdf com toda a documentação relativa à análise e design realizada e outro pdf com toda a documentação comprovativa da adoção do SCRUM.

O trabalho deverá ser submetido no Moodle até às 17:30 do dia 12 de março de 2021 (sexta-feira).