TRABALHO PRÁTICO Meta 2 - 2 valores

Pretende-se, com esta segunda meta, a integração de um **serviço** *Web* **do tipo REST (API REST)** no servidor principal desenvolvido na meta anterior (ver informação mais à frente sobre os casos em que o servidor principal não está operacional ou não foi desenvolvido), bem como uma aplicação cliente que o utilize. Para o efeito, deve ser criado um projeto Spring Boot do tipo Maven, onde é integrado o código fonte do servidor da primeira meta (se estiver operacional), e acrescentada a lógica relativa à API REST pretendida (controladores, autenticação, etc.).

No caso de trabalhos práticos com a primeira meta funcional, é aconselhável os controladores REST recorrerem à lógica de acesso à bases de dados já existente (por exemplo, uma classe do tipo *DBManager* se foi essa a opção na primeira meta). Também podem ser introduzidas, apesar de não serem alvo de avaliação, as alterações que os grupos entenderem serem necessárias ao que foi apresentado na primeira meta.

A aplicação cliente a desenvolver deve utilizar todas as funcionalidades oferecidas pela API REST pretendida e pode estar baseada na aplicação da primeira meta, mas com a interação com o servidor principal efetuada via pedidos HTTP/API REST em vez de *sockets* TCP (ver os exemplos de clientes de API REST apresentados nas aulas teóricas). A notificação assíncrona dos clientes deixa de ser um requisito.

Quando a primeira meta não está funcional ou não foi realizada, o projeto Spring Boot a desenvolver consiste apenas na API REST pretendida com acesso direto a uma base de dados SQLite já existente (previamente criada, por exemplo, através da aplicação SQLite Studio). Neste caso, não é necessário realizar as verificações exigidas na primeira meta quanto aos períodos de validade dos códigos gerados e submetidos. A estrutura da base de dados deve estar corretamente configurada para impedir tanto a inserção de entradas duplicadas nas tabelas como inconsistências resultantes da eliminação de determinadas entradas nas tabelas.

A API REST a desenvolver deve permitir:

- Registar um novo utilizador, sendo este caraterizado por um nome, um número de identificação, um endereço de email e uma password. O endereço de email deve ser único e serve de username;
- Autenticar um utilizador já registados através do respetivo endereço de email e password (autenticação básica HTTP com devolução de um token JWT);
- Verificar se o utilizador autenticado tem privilégios de administrador;
- Submeter o código associado a um evento para efeitos de registo da presença;
- Consultar as presenças registadas, podendo ser aplicados diversos tipos de critérios/filtros opcionais (período, nome do evento, etc.);

José Marinho 1/3

- Se o utilizador for um administrador:
 - Criar um evento, sendo este caracterizado pelo nome, local, data de realização, hora de início e hora de fim;
 - Eliminar um evento, desde que ainda não tenha qualquer presença registada;
 - Consultar os eventos criados, podendo ser aplicados diversos tipos de critérios/filtros opcionais (período, nome do evento, etc.);
 - Gerar um código de registo de presenças para um evento, com indicação do tempo de validade em minutos;
 - Consultar as presenças registadas num determinado evento.

Todas as funcionalidades, com exceção do registo e autenticação de utilizadores, podem apenas ser executadas por utilizadores previamente autenticados. Este requisito é garantido através da inclusão de um *token* JWT (devolvido numa autenticação bem-sucedida) no campo *Authorization* do cabeçalho dos pedidos HTTP (*Authorization*: *Bearer* < token JWT>).

Na autenticação, deve ser usado o mecanismo básico de autenticação HTTP, em que as credenciais são transmitidas, com codificação base64, no campo *Authorization* do cabeçalho do pedido (Authorization: Basic <credentials>). A resposta, caso a autenticação seja bem-sucedida, contém o *token* JWT que deve ser incluído nos pedidos subsequentes. Este deve ter um tempo de validade de 5 minutos a contar do momento em que é gerado.

Como o token JWT identifica univocamente um utilizador autenticado, a identidade de quem emite um pedido à API REST não deve constar da URI nem do corpo das mensagens HTTP. Caso o token não exista no pedido ou seja inválido, os pedidos HTTP devem devolver um código de resposta 401 (Unauthorized). Nas restantes situações em que um pedido não pode ser satisfeito com sucesso (por exemplo, lançamento de uma SQLException ou indicação de evento inexistente), deve ser devolvida uma resposta com um código de resposta HTTP apropriado e uma mensagem explicativa no seu corpo.

A aplicação cliente a desenvolver devem recorrer, entre outras, aos packages java.net, javax.json e com.google.gson (ver os exemplos apresentados nas aulas teóricas), não podendo estar baseada na plataforma Spring Boot. Quando esta não possui ou deixa de possuir um token JWT válido (o código de resposta aos pedidos é igual a 401), deve oferecer aos utilizadores apenas a possibilidade de registo ou login. As opções reservadas aos administradores, devem apenas ser apresentadas se o utilizador autenticado possuir esse perfil.

Observações

 Aconselha-se testar a API REST durante o seu desenvolvimento recorrendo à aplicação Postman ou ao comando curl. Só depois de esta estar validade é que é aconselhável desenvolver/adaptar a aplicação cliente.

José Marinho 2/3

- Para efeitos de desenvolvimento, teste e apresentação da API REST durante a defesa, a base de dados pode ser pré-preenchida de forma manual com dados (por exemplo, através do SQLite Studio).
- O relatório deve obrigatoriamente incluir uma descrição/documentação suficientemente clara e detalhada dos endpoints da API REST desenvolvida para que qualquer pessoa que pretenda usá-la consiga fazê-lo sem dificuldade e sem ter de analisar o código fonte. Este pode, por exemplo, incluir uma tabela de síntese semelhante à seguinte (com uma coluna adicional para eventuais parâmetros, opcionais ou não, nas URI) e, de seguida, uma descrição mais detalhada da estrutura dos corpos das mensagens de pedido e de resposta, quando existem (pode ser através de um exemplo concreto em JSON).

| Method | URI | Operation | Description | Request body | Response body |
|--------|------------------|-----------|---------------------------------|--------------|------------------------|
| GET | /api/movies | Read | Get all movie records | None | Array of movie records |
| GET | /api/movies/{id} | Read | Get a movie record by ID | None | Movie record |
| POST | /api/movies | Create | Add a new movie record | Movie record | Movie record |
| PUT | /api/movies/{id} | Update | Update an existing movie record | Movie record | None |
| DELETE | /api/movies/{id} | Delete | Delete a movie record | None | Movie record |

Considerações Gerais

Deve ter-se em consideração que esta segunda meta do trabalho:

- Encontra-se sujeita às mesmas condições gerais da primeira meta;
- Deverá ser entregue até ao dia 18 de dezembro de 2023, às 8h00, através da plataforma InforEstudante, num ficheiro ZIP com a designação PD-23-24-F2-TP-Gx.zip, sendo x o número do grupo;
- O ficheiro ZIP deve incluir o relatório, o código fonte do cliente e o projeto Sping Boot (sem a pasta out e com uma base de dados SQLite já povoada com alguns dados).

José Marinho 3/3