**Resposta** **Spring Boot API Rest: Segurança da API, Cache e Monitoramento**

Já temos muitos alunos e alunas cadastradas e agora temos muito acesso para visualizar os perfis. Além disso, temos um novo endpoint de listagem e essa lista só cresce. Para fechar, é mais do que importante a gente controlar o acesso a tais informações.

1  
Dado que as informações cadastradas dos alunos e alunas quase nunca muda, o que você faria para evitar que a recuperação dessa informação fosse feita sempre a partir do banco de dados?

* Usaria o recurso de cache, assim na primeira vez que for realizado uma chamada essa chamada vai ser executada no banco de dados vai devolver o resultado, mas nas próximas chamadas ele já devolve direto o que está em memória. Com isso, fica muito mais rápido o retorno para o cliente que fez a chamada para a API.

2  
Na listagem é importante trabalharmos com dados paginados. Descreva em detalhes os passos de implementação que você faria para possibilitar que a aplicação cliente pudesse acessar as informações de paginada e porque realizar a paginação é importante.

* Criar um método Get no controller que retornasse uma página de alunoDTO no caso Page<>, esse método vou chamar de getAllPage, ele recebe um parâmetro do tipo Pageable caso queira especificar algum parâmetro especifico como trazer por nome usamos a anotação @RequestParam para receber essa especificação via parâmetro da requisição.
* Criar a classe DTO pra que ira fazer a abstração da entity, e nela conter os atributos desejados que retorne e os getters, já que será apenas usado para pesquisa, criar um construtor de aluno como parâmetro para realizar os gettes para os atributos desejados e criar um método de conversão de entity para dto, que vai retornar uma Page de dto que será usada no controller, usando map para retornar um alunoDTO.
* Dentro de método criado no controller, usar o método já existente no repository findAll que tbm devolve uma page e retornar essa pagina recebida de aluno já convertida para dto com o método criado na classe dto como resposta no método, com os status 200 ok.
* Para não devolver todos os registros de uma vez nas requisições, num banco de dados muito grande acaba ficando lento carregar todo vez que recebemos uma requisição com todos os dados do banco, então conseguimos limitar a quantidade de retorno , ordenar como queremos receber esses dados tbm.

3  
Para fechar, descreva como funciona o mecanismo de autenticação e autorização para uma API Rest através de tokens.

* O cliente envia as credenciais ali a senha email para o aplicação, ela autentica esses dados e gera um token que é devolvido para o cliente, O cliente salva esse token e envia ele em um header em cada requisição, já que não usa a autenticação via session que guarda a autenticação em memória, toda requisição deve ser autenticada novamente com o token, a aplicação, em cada requisição extrai o token e verifica se o token é valido ou não, se o token é valido, aceita a requisição se não rejeita e devolve um status code 403, que não foi autorizado a requisição.

Através do token usamos a autenticação de maneira stateless

. O cliente envia as credenciais para o servidor.

2. O servidor autentica as credenciais e gera um token.

3. O servidor envia o token para o cliente.

4. O cliente salva esse token e envia ele em um header em cada requisição

6. O servidor, em cada requisição extrai o token e verifica se o token e valido ou não

- Se o token é valido, o servidor aceita a requisição

  - Se o token é inválido, o servidor rejeita a requisição

7. O servidor pode ter um endpoint que renova o token

Acabei não especificando que o token é criptografado e nem gerado pela JWT



**Spring Boot API REST: Construa uma API**

|  |
| --- |
| **Cenário:** |
| Já temos muitos alunos e alunas cadastradas e agora temos muito acesso para visualizar os perfis. Além disso, temos um novo endpoint de listagem e essa lista só cresce. Para fechar, é mais do que importante a gente controlar o acesso a tais informações.   * Dado que as informações cadastradas dos alunos e alunas quase nunca muda, o que você faria para evitar que a recuperação dessa informação fosse feita sempre a partir do banco de dados? * Na listagem é importante trabalharmos com dados paginados. Descreva em detalhes os passos de implementação que você faria para possibilitar que a aplicação cliente pudesse acessar as informações de paginada e porque realizar a paginação é importante. * Para fechar, descreva como funciona o mecanismo de autenticação e autorização para uma API Rest através de tokens. |

|  |
| --- |
| **O que seria bom ver nessa resposta?** |

* **Peso 3:**Utilização do Spring Cache para aumentar a velocidade de acesso às informações de detalhe dos(as) alunos(as).
* **Peso 2:**Detalhes que precisam aparecer para o uso da paginação
* **Peso 1:**O motivo de utilizar a paginação
* **Peso 0.25:**Método novo acessado via GET para listagem
* **Peso 0.25:**Parâmetro do tipo Pageable no método
* **Peso 0.25:**Criação de um método no repository já recebendo um Pageable como argumento e retornando um tipo que representa Paginaçao.
* **Peso 0.25:**Retorno do objeto do tipo que representa uma Paginaçao conectado com um outro objeto de saída, para controlar as informações certinhas.
* **Peso 5:**Na resposta sobre segurança é importante ver uma referência a utilização do token e também sobre o funcionamento do token.
* **Peso 2.5:** Explicação da utilização do token associada ao fato de não querermos manter estado no lado do servidor.
* **Peso 2.5:**Explicação sobre o conteúdo do token, informando que a informação ali é criptografada e que contém tudo que é necessário para verificar a existência do usuário na aplicação a cada nova requisição.

|  |
| --- |
| **Resposta do Especialista:** |
| * Para não acessar o banco de dados o tempo todo para visualizar os detalhes dos(as) alunos(as), eu posso configurar o cache dentro do projeto, através do Spring Cache. Minha estratégia seria começar anotando o método do controller de detalhe com a annotation de Cache. Um ponto de atenção seria a configuração de timeout dos dados em função da necessidade de negócio. * Para a paginação: * Paginamos os dados porque é importante controlar o número de informações retornados nos endpoints. Se a gente não controla, a performance pode ser afetada assim como a própria visualização dos dados. * Primeiro eu crio o método de listagem conectado com um endereço de acesso via GET * Defino um argumento do tipo Pageable para este método * Crio um método no Repository(crio o Repository caso não exista) que recebe como argumento o objeto do tipo Pageable * Transformo os objetos retornados pelo método do Repository para objetos de um tipo de saída para a listagem. * Retorno o objeto que representa a paginação a partir do método do controller. * Para fazer um sistema de autenticação e autorização numa API Rest é necessário fazer com que a informação que representa o "usuário logado" seja passada em cada requisição feita do cliente para o servidor. Chamamos essa informação de "token". Isso pode ser feito via cookie, header, parâmetro de requisição etc.  Isto é feito por conta da natureza stateless da API, a ideia é que toda informação venha sempre do cliente. Isso facilita qualquer ação relativa a escalabilidade futura.  Este token geralmente contém algumas informações do usuário criptografadas a partir de uma chave. Podemos colocar timeout também nesse token para controlar tempo de acesso. A especificação sugerida para guiar a criação do token é a JWT. O Spring Security suporta tudo isso através de configurações um tanto quanto extensas. |