Bootcamp Spring React 3.0 - Cap. 05

Consultas ao banco de dados

Competências

- SQL e JPQL
  - Revisão SQL, referências, exercícios
  - Estudos de caso SQL e JPQL
    - Projeção, restrição, escalares
    - Joins
    - Group by
    - União
    - Diferença
- Spring Data JPA
  - Query methods
  - Estudo de caso: busca detalhada com parâmetros opcionais e paginação
  - Problema N+1 consultas

SQL raiz

Revisão de Álgebra Relacional e SQL

https://www.youtube.com/watch?v=GHpE5xOxXXI

Referências

https://www.w3schools.com/sql/default.asp

Exercícios

https://urionlinejudge.com.br/

Grupo 1: projeção, restrição
2602, 2603, 2604, 2607, 2608, 2615, 2624

Grupo 2: JOIN
2605, 2606, 2611, 2613, 2614, 2617, 2618, 2619, 2620, 2621, 2622, 2623, 2742

Grupo 3: GROUP BY, subconsultas
2609, 2993, 2994, 2995, 2996

Grupo 4: Expressões na projeção
2610, 2625, 2738, 2739, 2741, 2743, 2744, 2745, 2746, 3001

Grupo 5: Diferença, União
2616, 2737, 2740, 2990

Grupo 6: Difíceis
2988, 2989, 2991, 2992, 2997, 2998, 2999

Estudos de caso: SQL e JPQL

Caso queira usar o servidor do Postgresql via Docker

```
docker run -p 5432:5432 --name meu-container-pg12 -e
POSTGRES_PASSWORD=1234567 -e POSTGRES_DB=minha_base postgres:12-alpine
```

Estudos de caso

- URI 2602 - busca simples
- URI 2611 - join simples
- URI 2621 - like e between
- URI 2609 - group by
- URI 2737 - união
- URI 2990 - diferença / left join

Spring Data JPA

Referências

https://docs.spring.io/spring-data/jpa/docs/current/reference/html

Aulas

- Query methods
- Estudo de caso: busca detalhada do DSCatalog
- Problema N+1 consultas

Figma do DSCatalog

https://www.figma.com/file/1n0aifcfatWv9ozp16XCrq/DSCatalog-Bootcamp

Disclaimer: avisos sobre o DSCatalog

Aviso 1: Amazon S3

Em alguns momentos da aula você vai ver alguns códigos referentes ao S3 da Amazon. Favor ignorá-los. Foque na construção da consulta JPQL. A parte do S3 foi para a coleção de conteúdos.

Aviso 2: Pageable

Nas aulas você vai ver a instanciação manual do Pageable com PageRequest, porém o código de referência atualizado (que está para download no material de apoio) já contém o Pageable simplificado conforme nós adequamos no capítulo 2:

Chamada no Postman

(antes) `{{host}}/products?page=0&linesPerPage=12&direction=ASC,orderBy=name`

(depois) `{{host}}/products?page=0&size=12&sort=name,asc`

ProductResource

```
@GetMapping
public ResponseEntity<Page<ProductDTO>> findAll(
		@RequestParam(value = "categoryId", defaultValue = "0") Long categoryId,
		@RequestParam(value = "name", defaultValue = "") String name,
		Pageable pageable) {

	Page<ProductDTO> list = service.findAllPaged(categoryId, name.trim(), pageable);

	return ResponseEntity.ok().body(list);
}
```

ProductService

```
@Transactional(readOnly = true)
public Page<ProductDTO> findAllPaged(Long categoryId, String name, Pageable pageable) {
	List<Category> categories = (categoryId == 0) ? null :
			Arrays.asList(categoryRepository.getOne(categoryId));
	Page<Product> list = repository.find(categories, name, pageable);
	return list.map(x -> new ProductDTO(x));
}
```

Configuração de CORS no DSCatalog

```
@Bean
public CorsConfigurationSource corsConfigurationSource() {
    CorsConfiguration corsConfig = new CorsConfiguration();
    corsConfig.setAllowedOriginPatterns(Arrays.asList("*"));
    corsConfig.setAllowedMethods(Arrays.asList("POST", "GET", "PUT", "DELETE", "PATCH"));
    corsConfig.setAllowCredentials(true);
    corsConfig.setAllowedHeaders(Arrays.asList("Authorization", "Content-Type"));

    UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();
    source.registerCorsConfiguration("/**", corsConfig);
    return source;
}

@Bean
public FilterRegistrationBean<CorsFilter> corsFilter() {
    FilterRegistrationBean<CorsFilter> bean
            = new FilterRegistrationBean<>(new CorsFilter(corsConfigurationSource()));
    bean.setOrder(Ordered.HIGHEST_PRECEDENCE);
    return bean;
}
```

Consulta de notificações do DSLearn

Figma do DSLearn

https://www.figma.com/file/p8Hawp1w5g0pCZ3h3ZsCUd/DSLearn-Bootcamp

```
@Query("SELECT obj FROM Notification obj WHERE "
        + "(obj.user = :user) AND "
        + "(:unreadOnly = false OR obj.read = false) "
        + "ORDER BY obj.moment DESC")
Page<Notification> find(User user, boolean unreadOnly, Pageable pageable);
```

Desafio para entregar

TAREFA: MovieFlix casos de uso

Implemente as funcionalidades necessárias para que os testes do projeto abaixo passem:
https://github.com/devsuperior/bds06

Collection do Postman:
https://www.getpostman.com/collections/72a46c64473b7611a021

ATENÇÃO (1): basta entregar a tarefa no banco H2. Não precisa configurar o Postgres no projeto, ok?

ATENÇÃO (2): se você tiver meio perdido pra começar, dê uma olhada na aula tira dúvidas que a gente fez com a turma 5: https://youtu.be/mulD7EShAGg

Visão geral do sistema MovieFlix

O sistema MovieFlix consiste em um banco de filmes, os quais podem ser listados e avaliados pelos usuários. Usuários podem ser visitantes (VISITOR) e membros (MEMBER). Apenas usuários membros podem inserir avaliações no sistema.

Ao acessar o sistema, o usuário deve fazer seu login. Apenas usuários logados podem navegar nos filmes. Logo após fazer o login, o usuário vai para a listagem de filmes, que mostra os filmes de forma paginada, ordenados alfabeticamente por título. O usuário pode filtrar os filmes por gênero.

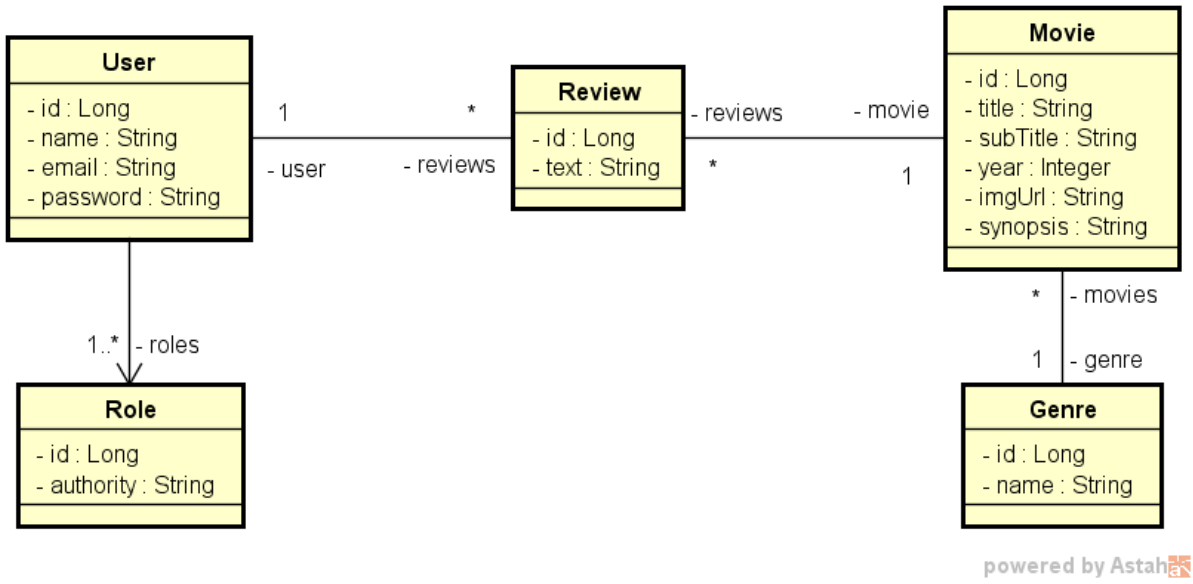
Ao selecionar um filme da listagem, é mostrada uma página de detalhes, onde é possível ver todas informações do filme, e também suas avaliações. Se o usuário for MEMBER, ele pode ainda registrar uma avaliação nessa tela.

Um usuário possui nome, email e senha, sendo que o email é seu nome de usuário. Cada filme possui um título, subtítulo, uma imagem, ano de lançamento, sinopse, e um gênero. Os usuários membros podem registrar avaliações para os filmes. Um mesmo usuário membro pode deixar mais de uma avaliação para o mesmo filme.

Protótipos de tela

https://www.figma.com/file/6JQVnxKgKtVHLleSBBgRin/MovieFlix-front-listagem

Modelo conceitual



Casos de uso

Efetuar login

1. [IN] O usuário **anônimo** informa seu email e senha
2. [OUT] O **sistema** informa um token válido

Listar filmes

1. [OUT] O **sistema** apresenta uma listagem dos nomes de todos gêneros, bem como uma listagem paginada com título, subtítulo, ano e imagem dos filmes, **ordenada alfabeticamente por título**.
2. [IN] O usuário **visitante ou membro** seleciona, opcionalmente, um gênero.
3. [OUT] O **sistema** apresenta a listagem atualizada, restringindo somente ao gênero selecionado.

Visualizar detalhes do filme

1. [IN] O usuário **visitante ou membro** seleciona um filme
2. [OUT] O **sistema** informa título, subtítulo, ano, imagem e sinopse do filme, e também uma listagem dos textos das avaliações daquele filme juntamente com nome do usuário que fez cada avaliação.
3. [IN] O usuário **membro** informa, opcionalmente, um texto para avaliação do filme.

4. [OUT] O sistema apresenta os dados atualizados, já aparecendo também a avaliação feita pelo usuário.

   Exceção 3.1 - Texto vazio
   3.1.1. O sistema apresenta uma mensagem de que não é permitido texto vazio na avaliação

Mínimo para aprovação*: 12/15
* Os testes de entidade foram mantidos no projeto para assegurar a correta estrutura do modelo de domínio, mas não entram na pontuação desta avaliação.