

Mudanças - Search com Specification

No nosso projeto Orbis, inicialmente implementamos uma busca utilizando JPQL estática. Esta abordagem consiste em uma query SQL fixa escrita manualmente que utiliza JOINS específicos para buscar eventos baseados em relacionamentos com as entidades Category e Tag. A query é definida diretamente na anotação `@Query` e sempre executa a mesma estrutura SQL, apenas variando os valores dos parâmetros. Embora seja funcional e adequada para casos específicos, essa abordagem apresenta limitações significativas em termos de flexibilidade: cada nova combinação de filtros de busca exigiria a criação de um novo método no repositório, levando ao problema conhecido como 'query explosion' - se tivéssemos 5 campos de filtro opcionais, precisaríamos potencialmente de 32 métodos diferentes para cobrir todas as combinações possíveis.

Para atender ao requisito específico do professor de implementar consultas dinâmicas utilizando Spring Data, avaliamos duas opções: Criteria API pura ou Specifications. Decidimos implementar Specifications ao invés da Criteria API diretamente por várias razões técnicas e práticas. As Specifications são uma abstração de alto nível construída sobre a Criteria API que oferece uma sintaxe mais limpa e expressiva, permitindo a composição de filtros de forma modular através de métodos como `.and()`, `.or()` e `.where()`. Cada Specification representa uma condição de busca individual e pode ser reutilizada em diferentes contextos, facilitando a manutenção e testabilidade do código.

Enquanto a Criteria API pura exige a escrita de código mais verboso com manipulação direta de CriteriaBuilder, Root e Predicate, as Specifications encapsulam essa complexidade em métodos menores e mais focados. Por exemplo, nossa Specification `hasTitle()` encapsula toda a lógica necessária para fazer uma busca case-insensitive por título, incluindo a verificação de valores nulos e a construção do predicado LIKE. Isso resulta em código mais legível, manutenível e menos propenso a erros.

Com a implementação de Specifications, conseguimos criar um sistema de busca verdadeiramente dinâmico onde o usuário pode filtrar eventos por qualquer combinação de título, localização e intervalo de datas. O método `searchEventsWithFilters()` constrói a query final baseada apenas nos parâmetros fornecidos, ignorando automaticamente os filtros que não foram especificados. Isso resolve elegantemente o problema do 'IF hell' que ocorreria se tentássemos implementar todas as combinações possíveis através de métodos estáticos, demonstrando a evolução natural da arquitetura do projeto em direção a soluções mais escaláveis e flexíveis.

Mudanças Implementadas no Código

Criação da classe EventSpecifications

Arquivo criado: `src/main/java/br/com/orbis/Orbis/service/EventSpecifications.java`

```
package br.com.orbis.Orbis.service;

import br.com.orbis.Orbis.model.Event;
import org.springframework.data.jpa.domain.Specification;
import java.time.LocalDate;

public class EventSpecifications {

    public static Specification<Event> hasTitle(String title) {
        return (root, query, criteriaBuilder) -> {
            if (title == null || title.trim().isEmpty()) {
                return null;
            }
        };
    }
}
```

```

    }
    return criteriaBuilder.like(
        criteriaBuilder.lower(root.get("title")),
        "%" + title.toLowerCase() + "%"
    );
};
}

public static Specification<Event> hasLocation(String location) {
    return (root, query, criteriaBuilder) -> {
        if (location == null || location.trim().isEmpty()) {
            return null;
        }
        return criteriaBuilder.like(
            criteriaBuilder.lower(root.get("location")),
            "%" + location.toLowerCase() + "%"
        );
    };
}

public static Specification<Event> isAfterDate(LocalDate date) {
    return (root, query, criteriaBuilder) -> {
        if (date == null) {
            return null;
        }
        return criteriaBuilder.greaterThanOrEqualTo(root.get("date"), date);
    };
}

public static Specification<Event> isBeforeDate(LocalDate date) {
    return (root, query, criteriaBuilder) -> {
        if (date == null) {
            return null;
        }
        return criteriaBuilder.lessThanOrEqualTo(root.get("date"), date);
    };
}
}
}

```

2. Modificação do EventRepository

Arquivo modificado: `src/main/java/br/com/orbis/Orbis/repository/EventRepository.java`

Mudança: Adicionada a interface *JpaSpecificationExecutor<Event>* para suporte a Specifications

```

@Repository
public interface EventRepository extends JpaRepository<Event, Long>,
    JpaSpecificationExecutor<Event> {

```

3. Adição de método no EventService

Arquivo modificado: `src/main/java/br/com/orbis/Orbis/service/EventService.java`

Imports adicionados:

```
import org.springframework.data.jpa.domain.Specification;
import java.time.LocalDate;
```

Método adicionado:

```
public List<Event> searchEventsWithFilters(String title, String location,
LocalDate startDate, LocalDate endDate) {
    Specification<Event> spec = Specification
        .where(EventSpecifications.hasTitle(title))
        .and(EventSpecifications.hasLocation(location))
        .and(EventSpecifications.isAfterDate(startDate))
        .and(EventSpecifications.isBeforeDate(endDate));

    return repository.findAll(spec);
}
```

4. Adição de endpoint no EventController

Arquivo modificado: `src/main/java/br/com/orbis/Orbis/controller/EventController.java`

Imports adicionados:

```
import org.springframework.format.annotation.DateTimeFormat;
import java.time.LocalDate;
```

Método adicionado:

```
@GetMapping("/search-dynamic")
public ResponseEntity<List<Event>> searchEventsWithFilters(
    @RequestParam(required = false) String title,
    @RequestParam(required = false) String location,
    @RequestParam(required = false) @DateTimeFormat(iso =
DateTimeFormat.ISO.DATE) LocalDate startDate,
    @RequestParam(required = false) @DateTimeFormat(iso =
DateTimeFormat.ISO.DATE) LocalDate endDate) {

    List<Event> events = service.searchEventsWithFilters(title, location,
startDate, endDate);
    return ResponseEntity.ok(events);
}
```

Resumo das Alterações

Arquivos criados: 1 (EventSpecifications.java)

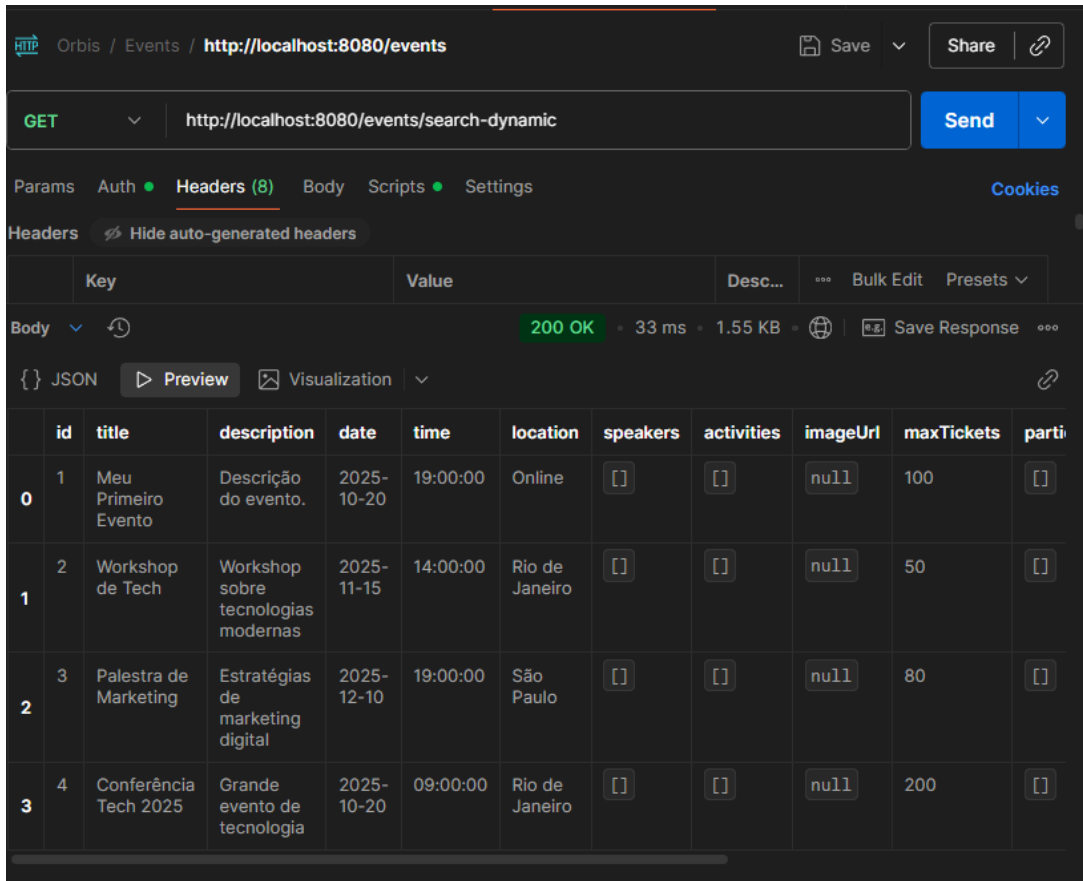
Arquivos modificados: 3 (EventRepository.java, EventService.java, EventController.java)

Resumo da funcionalidade adicionada: Sistema de busca dinâmica que permite filtrar eventos por qualquer combinação de título, localização e intervalo de datas através do endpoint

/events/search-dynamic

Testando a nova funcionalidade:

Sem filtro nenhum:



HTTP Orbis / Events / <http://localhost:8080/events> Save Share

GET <http://localhost:8080/events/search-dynamic> Send

Params Auth Headers (8) Body Scripts Settings Cookies

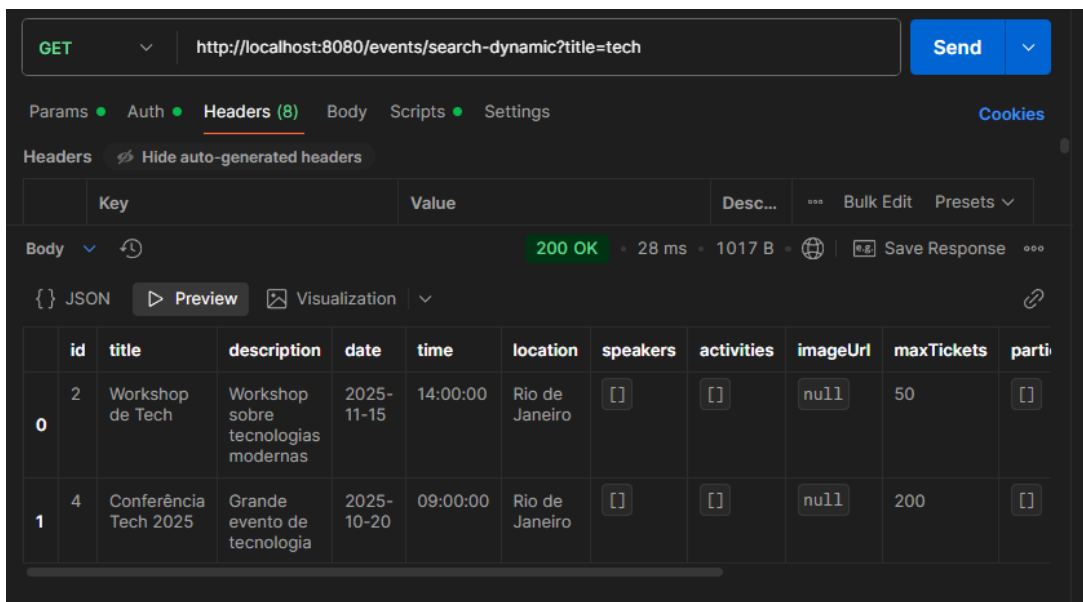
Headers Hide auto-generated headers

Body 200 OK 33 ms 1.55 KB Save Response

JSON Preview Visualization

	id	title	description	date	time	location	speakers	activities	imageUrl	maxTickets	parti
0	1	Meu Primeiro Evento	Descrição do evento.	2025-10-20	19:00:00	Online	[]	[]	null	100	[]
1	2	Workshop de Tech	Workshop sobre tecnologias modernas	2025-11-15	14:00:00	Rio de Janeiro	[]	[]	null	50	[]
2	3	Palestra de Marketing	Estratégias de marketing digital	2025-12-10	19:00:00	São Paulo	[]	[]	null	80	[]
3	4	Conferência Tech 2025	Grande evento de tecnologia	2025-10-20	09:00:00	Rio de Janeiro	[]	[]	null	200	[]

Buscar por título



GET <http://localhost:8080/events/search-dynamic?title=tech> Send

Params Auth Headers (8) Body Scripts Settings Cookies

Headers Hide auto-generated headers

Body 200 OK 28 ms 1017 B Save Response

JSON Preview Visualization

	id	title	description	date	time	location	speakers	activities	imageUrl	maxTickets	parti
0	2	Workshop de Tech	Workshop sobre tecnologias modernas	2025-11-15	14:00:00	Rio de Janeiro	[]	[]	null	50	[]
1	4	Conferência Tech 2025	Grande evento de tecnologia	2025-10-20	09:00:00	Rio de Janeiro	[]	[]	null	200	[]

Buscar por localização

GET ▼ `http://localhost:8080/events/search-dynamic?location=rio` Send ▼

Params ● Auth ● **Headers (8)** Body Scripts ● Settings Cookies

Headers Hide auto-generated headers

	Key	Value	Desc...	...	Bulk Edit	Presets	▼
--	-----	-------	---------	-----	-----------	---------	---

Body ▼ 🔄 200 OK 15 ms 1017 B 🌐 📄 Save Response ...

{} JSON ▶ Preview 🖼 Visualization ▼ 🔗

	id	title	description	date	time	location	speakers	activities	imageUrl	maxTickets	parti
0	2	Workshop de Tech	Workshop sobre tecnologias modernas	2025-11-15	14:00:00	Rio de Janeiro	⌂	⌂	null	50	⌂
	4	Conferência Tech 2025	Grande evento de tecnologia	2025-10-20	09:00:00	Rio de Janeiro	⌂	⌂	null	200	⌂

Buscar por intervalo de datas

GET ▼ `http://localhost:8080/events/search-dynamic?startDate=2025-11-01&endDate=2025-12-31` Send ▼

Params ● Authorization ● **Headers (8)** Body Scripts ● Settings Cookies

Headers Hide auto-generated headers

	Key	Value	Descript...	...	Bulk Edit	Presets	▼
--	-----	-------	-------------	-----	-----------	---------	---

Body ▼ 🔄 200 OK 17 ms 1016 B 🌐 📄 Save Response ...

{} JSON ▶ Preview 🖼 Visualization ▼ 🔗

	id	title	description	date	time	location	speakers	activities	imageUrl	maxTickets	participants	b
0	2	Workshop de Tech	Workshop sobre tecnologias modernas	2025-11-15	14:00:00	Rio de Janeiro	⌂	⌂	null	50	⌂	1
	3	Palestra de Marketing	Estratégias de marketing digital	2025-12-10	19:00:00	São Paulo	⌂	⌂	null	80	⌂	7

Combinação de filtros

GET ▼ `http://localhost:8080/events/search-dynamic?title=tech&location=rio` Send ▼

Params ● Auth ● **Headers (8)** Body Scripts ● Settings Cookies

Headers Hide auto-generated headers

	Key	Value	Desc...	...	Bulk Edit	Presets	▼
--	-----	-------	---------	-----	-----------	---------	---

Body ▼ 🔄 200 OK 16 ms 1017 B 🌐 📄 Save Response ...

{} JSON ▶ Preview 🖼 Visualization ▼ 🔗

	id	title	description	date	time	location	speakers	activities	imageUrl	maxTickets	parti
0	2	Workshop de Tech	Workshop sobre tecnologias modernas	2025-11-15	14:00:00	Rio de Janeiro	⌂	⌂	null	50	⌂
	4	Conferência Tech 2025	Grande evento de tecnologia	2025-10-20	09:00:00	Rio de Janeiro	⌂	⌂	null	200	⌂