Refactorização dos Serviços de Procura

Em baixo é apresentado o código de ambos os serviços de procura entregues na 3ª entrega.

Procura prato por subnome:

```
public class ProcurarPratoService extends
        RestService {

    private final List<PratoDto> resultado = new ArrayList<PratoDto>();
    private final PratoDto pratoDto;


    public ProcurarPratoService(PratoDto prato) {
        this.pratoDto = prato;
    }

    /**
     * Procura pratos no portal que correspondam a um nome específico.
     * @see pt.ist.rest.service.RestService#dispatch()
     */
    @Override
    public final void dispatch() {

        PortalDeRestaurantes portal = FenixFramework.getRoot();


        for (Restaurante r : portal.getRestauranteSet()) {
            for (Prato p : r.getPrato()) {

                final String subnome = pratoDto.getNome();
                final String nomePrato = p.getNome();

                if (nomePrato.contains(subnome)) {
                    resultado.add(criaPratoDto(p));
                }
            }
        }

    }

    /**
     * @return resultado lista de pratos que correspondem ao nome pedido.
     */
    public List<PratoDto> getResultado() {
        return this.resultado;
    }


    private PratoDto criaPratoDto(Prato p) {
        //@formatter:off
        return new PratoDto(p.getNome(), p.getId(), p.getPreco(), p.getCalorias(),
                p.getRestaurante().getNome());
        //@formatter:on
    }
}
```

Procura prato por tipo:

```
public class ProcurarPratosTipoService extends
        RestService {

    private String tipo;
```

```
    private Collection<PratoDto> pratos;

    public ProcurarPratosTipoService(String tipo) {
        this.tipo = tipo;
        this.pratos = new ArrayList<PratoDto>();
    }

    /**
     * Procura pratos de um tipo específico no Portal.
     *
     * @throws NoSuchTypeException caso seja dado um tipo inexistente.
     * @see pt.ist.rest.service.RestService#dispatch()
     */
    @Override
    protected void dispatch() throws NoSuchTypeException {
        final PortalDeRestaurantes portal = FenixFramework.getRoot();

        if (!(tipo.equals("Vegetariano") || tipo.equals("Peixe") ||
tipo.equals("Carne"))) {
            throw new NoSuchTypeException(tipo);
        }
        for (Restaurante r : portal.getRestaurante()) {
            for (Prato p : r.getPrato()) {
                if (p.getTipo().equals(tipo))
                    pratos.add(new PratoDto(p.getNome(), p.getId(), p.getPreco(),
p.getCalorias(),
                            p.getRestaurante().getNome()));
            }
        }
    }

    /**
     * @return pratos coleção de pratos que correspondem ao tipo pedido.
     */
    public Collection<PratoDto> getPratos() {
        return pratos;
    }
}
```

As duas classes em cima foram reimplementadas na classe ProcuraService cujo código se mostra em baixo.

Código do Serviço Final:

```
public class ProcuraService extends
        RestService {

    private ArrayList<SearchDto> search;

    private SearchComponent sc;

    private ArrayList<PratoDto> resultado = new ArrayList<PratoDto>();

    public ProcuraService(ArrayList<SearchDto> procura) {
        this.search = procura;

    }

    @Override
    public void dispatch() {

        PortalDeRestaurantes pdr = FenixFramework.getRoot();

        sc = new SearchComponent(pdr);
        for (SearchDto sD : this.search) {
            switch (SearchEnum.fromValue(sD.getEnum())) {
```

```
                case NOME:
                    ProcurarPratoNome ppn = new ProcurarPratoNome(sD.getArg());
                    ppn.searchPrato(sc);
                    break;
                case TIPO:
                    ProcurarPratosTipo ppt = new ProcurarPratosTipo(sD.getArg());
                    ppt.searchPrato(sc);
                    break;
                default:
                    System.out.println("nao reconhece o tipo " + sD.getEnum());
                    break;
            }
        }
        for (Prato p : sc.getPratos()) {
            resultado.add(new PratoDto(p.getNome(), p.getId(), p.getPreco(),
p.getCalorias(), p
                    .getRestaurante().getNome()));
        }
    }

    public ArrayList<PratoDto> getResultado() {
        return resultado;
    }

}
```

Como se pode verificar, inicialmente os serviços de pesquisa usavam um grande bloco de código semelhante sendo os dois ciclos que iteravam sobre os restaurantes e os pratos em tudo iguais e o *if* condicional a grande diferença.

Apesar disso, era importante perceber a diferença entre uma pesquisa por subnome e uma pesquisa por tipo porque, visto que ambos os serviços recebem um argumento do tipo *String,* seria fácil confundi-los caso se fizesse uma pesquisa por um subnome que fosse, por exemplo, “Peixe” ou “Carne”.

A solução que implementámos consiste em utilizar uma classe auxiliar *SearchComponent* que irá guardar o resultado das procuras, sendo esta constituída inicialmente por todos os pratos de todos os restaurantes.

Além desta classe a solução utiliza também enumerados de forma a que, para adicionar novas procuras, seja apenas necessário adicionar o novo tipo de procura no enumerado e uma nova classe, responsável pelo novo tipo de procura, que irá herdar da classe *ProcurarPrato.* Desta forma para procurar pratos, basta chamar o serviço com uma lista de *SearchDto*’s que irão verificar os tipos com o enumerado acima descrito e chamar a procura respectiva.

Isto torna a criação de novos tipos de procura extremamente fácil e, até, intuitiva.