Código antigo:

```
public class ProcurarPratosNomeService extends PortalRestauranteService{

    private List<PratoDto> listaPratos = new ArrayList<PratoDto>();
    private String nomePrato;

    public ProcurarPratosNomeService(String nome){
        this.nomePrato = nome;
    }

    @Override
    public void dispatch() {

        PortalRestaurante pr = FenixFramework.getRoot();

        for(Restaurante r : pr.getRestaurante())
            for(Prato p : r.getPrato())

                if(p.getNome().contains(nomePrato)){
                    PratoDto pDto = new PratoDto(p.getNome(), p.getPreco(), p.getCalorias(),
                                                 p.getId(), p.getClassificacao(),
                                                 new RestauranteDto(r.getNome(), r.getMorada(),
                                                                    r.getClassificacao()));
                    listaPratos.add(pDto);
                }
    }

    public List<PratoDto> getResult(){
        return listaPratos;
    }
}
```

```
public class ProcurarPratosTipoService extends PortalRestauranteService{

    private AlimentoDto alimento;
    private List<PratoDto> listaPratos = new ArrayList<PratoDto>();

    public ProcurarPratosTipoService(AlimentoDto a){
        this.alimento = a;
    }

    @Override
    public void dispatch() {
        PortalRestaurante portal = FenixFramework.getRoot();


        for(Restaurante r : portal.getRestaurante())
            for(Prato p : r.getPrato())
                if(alimento.getTipo().equals("vegetariano")){
                    if(p.vegetariano())
                        listaPratos.add(new PratoDto(p.getNome(), p.getPreco(),
                                        p.getCalorias(), p.getId(), p.getClassificacao(),
                                        new RestauranteDto(r.getNome())));

                }else if(alimento.getTipo().equals("carne")){
                    if(p.tipoCarne())
                        listaPratos.add(new PratoDto(p.getNome(), p.getPreco(),
                                        p.getCalorias(), p.getId(), p.getClassificacao(),
                                        new RestauranteDto(r.getNome())));

                }else {
                    if(p.tipoPeixe())
                        listaPratos.add(new PratoDto(p.getNome(), p.getPreco(),
                                        p.getCalorias(), p.getId(), p.getClassificacao(),
                                        new RestauranteDto(r.getNome())));
                }
    }

        public List<PratoDto> getResult(){
            return listaPratos;
        }
    }
```

Código novo:

```
public abstract class ProcuraPratosService extends PortalRestauranteService {
    protected List<PratoDto> listaPratos = new ArrayList<PratoDto>();

    @Override
    public void dispatch() throws RestException {
        PortalRestaurante pr = FenixFramework.getRoot();

        for(Restaurante r : pr.getRestaurante())
            for(Prato p : r.getPrato())
                procuraPrato(p, r);
    }

    public abstract void procuraPrato(Prato prato, Restaurante r);

    public List<PratoDto> getResult(){
        return listaPratos;
    }
}
```

```
public class ProcurarPratosTipoService extends ProcuraPratosService{

    private AlimentoDto alimento;

    public ProcurarPratosTipoService(AlimentoDto a){
        this.alimento = a;
    }

    @Override
    public void procuraPrato(Prato p, Restaurante r) {
        if(alimento.getTipo().equals("vegetariano")){
            if(p.vegetariano())
                listaPratos.add(new PratoDto(p.getNome(), p.getPreco(), p.getCalorias(), p.getId(),
                                p.getClassificacao(), new RestauranteDto(r.getNome())));

        }else if(alimento.getTipo().equals("carne")){
            if(p.tipoCarne())
                listaPratos.add(new PratoDto(p.getNome(), p.getPreco(), p.getCalorias(), p.getId(),
                                p.getClassificacao(), new RestauranteDto(r.getNome())));

        }else {
            if(p.tipoPeixe())
                listaPratos.add(new PratoDto(p.getNome(), p.getPreco(), p.getCalorias(), p.getId(),
                                p.getClassificacao(), new RestauranteDto(r.getNome())));
        }
    }
}
```

```
public class ProcurarPratosNomeService extends ProcuraPratosService{

    private String nomePrato;

    public ProcurarPratosNomeService(String nome){
        this.nomePrato = nome;
    }

    @Override
    public void procuraPrato(Prato p, Restaurante r) {
        if(p.getNome().contains(nomePrato)){
            PratoDto pDto = new PratoDto(p.getNome(), p.getPreco(), p.getCalorias(), p.getId(), p.getClassificacao(),
                                         new RestauranteDto(r.getNome(), r.getMorada(), r.getClassificacao()));
            listaPratos.add(pDto);
        }
    }
}
```

De modo a efectuar a refactorização pedida, optámos por implementar o padrão de desenho Template Method.

Desta forma, abstraímos o código repetido para a super-classe ProcuraPratosService.

De seguida, criou-se o método abstracto procuraPrato que é especificado nas sub-classes.

Esta refactorização permitiu:

- A implementação de um método abstracto, em várias sub-classes, com comportamento variável.
- Evitamos replicação do código: o código comum é implementado na classe abstracta e o código que varia é implementado em cada sub-classe.