Nome**: Gabriel Maia** Matrícula: **1502979**

Gerenciamento de ordens de serviço

Arquitetura de referências para negócios

Curitiba 16 de junho de 2020

Introdução

O Projeto apresentado neste documento tem como foco principal gerenciar ordens de serviço. Durante a implementação do projeto houve-se a necessidade de modificar o diagrama de classes inicial, para atender aos padrões de projeto e algumas melhorias identificadas durante o desenvolvimento. Os padrões aplicados ao sistema são: Prototype e chain of responsibility.

Protype:

Este pattern foi introduzido nas classes de entidade Order, Request e User. Escolhi o uso deste pattern pela facilidade com que posso clonar os objetos tanto para replica-los em tela, quanto copiar itens do banco de dados.

Código:

```
0 referências
public Order Clone()
{
    Order clone = this.MemberwiseClone() as Order;
    clone.Id = this.Id;
    clone.Description = this.Description;
    clone.Queue = this.Queue;
    clone.Attachment = this.Attachment;
    clone.Request = this.Request.Clone();
    clone.User = this.User.Clone();
    return clone;
}
```

```
1 referência
public Request Clone(){
    Request request = this.MemberwiseClone() as Request;
    request.Id = this.Id;
    request.Equipament = this.Equipament;
    request.isDptoPayment = this.isDptoPayment;
    request.Scheduling = this.Scheduling;
    request.Status = this.Status;
    request.TechnicianDescription = this.TechnicianDescription;
    request.Type = this.Type;
    request.User = this.User.Clone();
    request.Approval = this.Approval;
    request.Description = this.Description;
    request.DescriptionDeclineApproval = this.DescriptionDeclineApproval;
    request.DescriptionsSupport =this.DescriptionsSupport;

    return request;
}
```

```
2 referencias
public User Clone()
{
    User user = this.MemberwiseClone() as User;
    user.AccessFailedCount = this.AccessFailedCount;
    user.ConcurrencyStamp = this.ConcurrencyStamp;
    user.Email = this.Email;
    user.EmailConfirmed = this.EmailConfirmed;
    user.First_Name = this.First_Name;
    user.Last_name = this.Last_name;
    user.LockoutEnabled = this.LockoutEnabled;
    user.LockoutEnd = this.LockoutEnd;
    user.Login = this.Login;
    user.NormalizedEmail = this.NormalizedEmail;
    user.NormalizedUserName = this.NormalizedUserName;
    user.PasswordHash = this.PasswordHash;
    user.PhoneNumber = this.PhoneNumber;
    user.PhoneNumberConfirmed = this.PhoneNumberConfirmed;
    user.SecurityStamp = this.SecurityStamp;
    user.TwoFactorEnabled = this.TwoFactorEnabled;
    user.UserName = this.UserName;

    return user;
}
```

chain of responsibility:

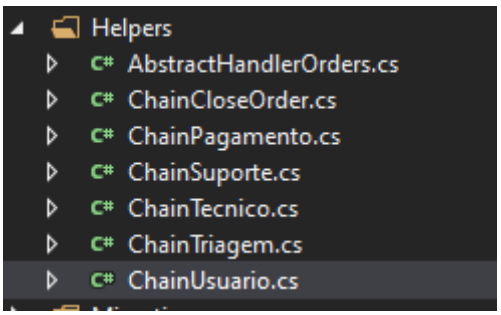
Este pattern foi escolhido para direcionar as ordens de serviço para usuários específicos, facilitando a responsabilidade de encaminhar para cada usuário. Para cada “step” que o usuário seleciona é escolhido um responsável diferente para realizar ação necessária.

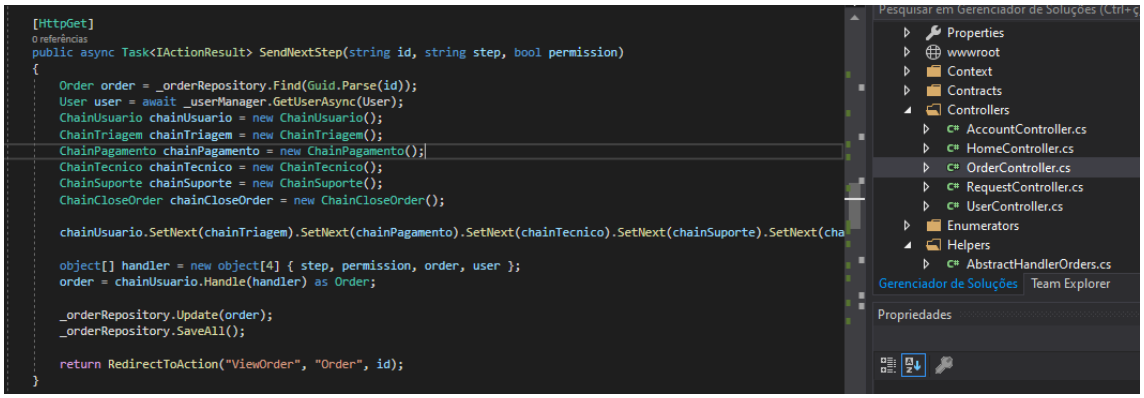
Código

```
6 referências
public interface IHandler
{
    6 referências
    public IHandler SetNext(IHandler handler);
    15 referências
    public object Handle(object[] request);
}
```

```
6 referências
public abstract class AbstractHandlerOrders : IHandler
{
    private IHandler _nextHandler;

    6 referências
    public IHandler SetNext(IHandler handler)
    {
        this._nextHandler = handler;
        return handler;
    }

    15 referências
    public virtual object Handle(object[] request)
    {
        if (this._nextHandler != null)
        {
            return this._nextHandler.Handle(request);
        }
        else
        {
```

```
public class ChainUsuario : AbstractHandlerOrders
{
    15 referências
    public override object Handle(object[] request)
    {
        if ((request[0] as String).Equals("usuario") && (request[1] == null ? false : true))
        {
            User user = request[3] as User;
            Order order = request[2] as Order;
            order.Queue = Enumerators.Queue.Requisitante;
            order.Request.Status = Enumerators.Status.Atualizado;
            order.Request.User = user;

            return order;
        }else
        {
            return base.Handle(request);
        }
    }
}
```





Diagramas:

Os diagramas podem ser encontrados através do endereço:
https://github.com/gabriel2mm/Gerencimento-OS

Dentro deste repositório, existe uma pasta chama “documentação”.