```
public class ProductService : SimpleService<Product>, ISimpleService<Product>
{
    public ProductService(IServiceProvider provider)
        : base(provider)
    {
        RuleFor(t => t.Title).Required().UniqAsync("A product has been defined with this title");
        RuleFor(t => t.Price).CustomValue(t => t < 0, "The product price cannot be negative");
        RuleFor(t => t.TakeoutPrice).CustomValue(t => t < 0, "The take out price cannot be negative");
        RuleFor(t => t.Meal).CustomValue(t => t == 0, "At least one meal must be selected");
        RuleFor(t => t.Code).UniqAsync("There is a product with this code in the system")
            .CustomValue(code =>
            {
                if (!code.HasValue())
                    return false;
                return code.Length < 3;
            }, "The product code must be at least three digits long")
            .Custom(p =>
            {
                if (!p.Code.HasValue())
                    return false;
                return new ProductCategoryService(ServiceProvider).GetAll().Any(pc => pc.Code == p.Code);
            }, "There is a product category with this code in the system");
    }


    public async Task UpdatePrice(int id, int price)
    {
        var old = await SingleAsync(id);
        old.Price = price;
        await UpdateAsync(old);
    }

    public async Task UpdateTakeoutPrice(int id, int takeoutPrice)
    {
        var old = await SingleAsync(id);
        old.TakeoutPrice = takeoutPrice;
        await UpdateAsync(old);
    }

    public async Task ToggleStatusAsync(int id)
    {
        var old = await SingleAsync(id);
        old.ActiveType = old.ActiveType == ActiveType.Enable ? ActiveType.Disable : ActiveType.Enable;
        await UpdateAsync(old);
    }

    public async Task ToggleOutofstock(int id)
    {
        var old = await SingleAsync(id);
        old.OutofStock = !old.OutofStock;
        await UpdateAsync(old);
    }
}
```