**Introdução ao Projeto – Levantamento de Requisito**

Como em qualquer projeto de software é necessário primeiramente começar pelo levantamento de requisitos, como nesse caso o requisito foi proposto pela Dextra então o segundo passo é a arquitetura de software.

**Arquitetura e Metodologia de Desenvolvimento**

Como arquitetura para esse software foi utilizados os padrões do ***SOLID*** e não haveria melhor forma de se conseguir isso sem ser utilizar a metodologia de desenvolvimento dirigida pelo domínio, chamado de ***DDD***, com essa metodologia é possível garantir os princípios do ***SOLID*** e assim deixar o código mais performático e o mais importante deixar mais fácil a manutenção e garantir a regra de negócio que fica na camada de Serviço.

Dentro do domínio foi utilizada a metodologia de desenvolvimento dirigido por testes, chamado de **TDD**, antes de se fazer a regra são feito os testes simulando situações que deveriam retornar resultados, como o método não foi implementado ainda nenhum teste passa e no momento que vão sendo feitas as implementações os testes começam passar, garantindo assim a regra agora, e futuramente quando houver novas implementações.

**Camada de Domínio e Acesso a Dados**

O software foi criado em ***C#***, primeiramente foi criada a camada de domínio onde foram criadas todas as interfaces que deveriam ser implementadas, Depois foram criados serviços, mas sem ter suas interfaces implementadas, depois foi criada a camada de acesso a dados utilizando o ***EntityFramework*** e ***DBContext***, pois ele traz uma facilidade grande para o gerenciamento de conexões e transações com o banco de dados, sem contar que é possível uma maior abstração do código, pois podemos criar classes genéricas que atendem a todas as entidades.

**Testes Automatizados**

Após a criação da camada de dados foram criados os testes automatizados como a biblioteca nativa do C# ***UnitTesting*** para verificar se a camada de serviço estava funcionando. Primeiramente todos os testes reprovaram, então após a implementação os testes passam, garantindo assim a regra.

Para garantir que os testes irão sempre passar indiferentemente do valor do preço ou quantidade que tenha no banco para cada produto no lanche, foram ***mockados*** os repositórios utilizando a biblioteca ***Moq***.

**Camada de aplicação**

Para garantir o principio da responsabilidade única foi criada na camada de aplicação as ***ViewModels*** que são responsáveis por trazer os dados no formato necessário para apresentar na tela.

E para fazer a conversão destes objetos de forma prática e rápida foi utilizado o ***AutoMapper*** que foi configurado na camada de apresentação.

**Camada de apresentação**

Para a camada de apresentação foi criado um projeto ***Asp.Net MVC*** que utiliza a metodologia ***MVC***, garantindo grande abstração no código e um aplicação mais leve para o usuário final. A interface do usuário possui elementos de ***HTML5***, o calculo do lanche é feito por uma chamada ***Ajax***, possui nela código ***JavaScript*** e ***Jquery***. Para customizar a página foi utilizado ***CSS*** e principalmente ***BootStrap*** que agiliza o desenvolvimento com elementos complexos e leves para uma aplicação ***MVC***.

Na camada de aplicação existe também a configuração do ***AutoMapper*** e do ***SimpleInjector***, que é responsável pela inversão de controle, garantindo assim a abstração de cada camada. Para que não houvesse a necessidade da camada de apresentação ter acesso a camada de dados foi criada a camada de ***CrossCutting*** onde fica a configuração do ***SimpleInjector***.

**Servidor de aplicação e Docker**

Foi criado o servidor de aplicação na minha maquina local utilizando ***IIS*** da Microsoft onde a aplicação funcionou perfeitamente, porém não foi possível realizar a criação da imagem via ***Docker***, pois ele roda em cima do ***Hyper-V*** e minha maquina é ***Windows 10 Home Edition***. Porém caso conseguisse instalar enfrentaria outras dificuldades pois o ***Docker*** está mais preparado para sistemas operacionais ***Linux*** com ***Apache*** como servidor de aplicação, consegui fazer simulações do ***Docker*** na minha maquina virtual porém funcionou apenas com aplicações ***Php*** com ***MySql*** como do ***WordPress***.