Projeto NOSSA LOJA

O projeto foi desenvolvido em Java utilizando o padrão arquitetural MVC.

MVC é um padrão de desenvolvimento de software que se adequa a esse tipo de projeto Web, pois permite a separação em camadas de regras de negócio e banco de dados, controladores das operações e interface de usuários, tudo isso visando o baixo acoplamento e alta coesão além de facilitar a manutenção e o reaproveitamento de código.

Foi utilizado na camada View a construção das páginas JSP que renderiza a página HTML no servidor antes de enviá-la ao navegador.

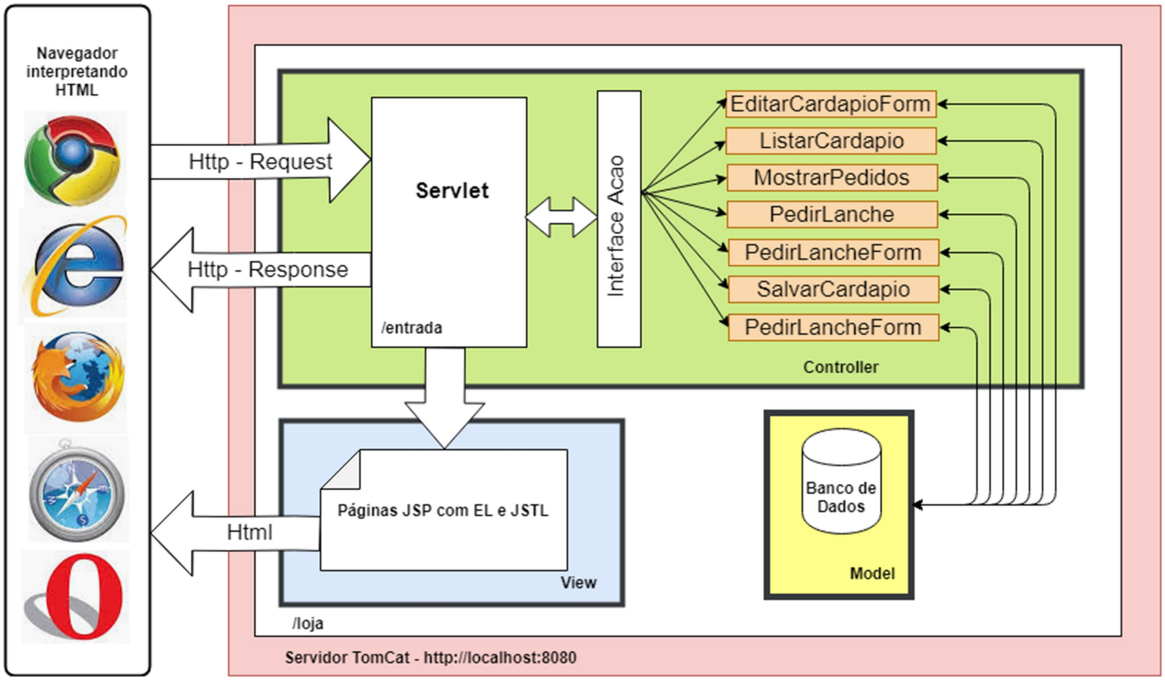
Utilizamos Expression Language (EL , ${ }) para executar a transferência dos atributos como lanche, ingredientes etc, lá do backend para o FrontEnd.

Para facilitar a execução de laços e desvios condicionais dentro da página JSP, utilizamos uma biblioteca JSTL (Java Standard Tag Library) que é uma biblioteca de tags para facilitar a “escrita” de código Java dentro de uma página JSP.

A abordagem do backend foi escolhida ser feita com Servlets, pois é a tecnologia que serve de base para outros frameworks mais utilizados como Spring MVC, VRaptor entre outros. Conhecendo bem essa base de funcionamento, a curva de aprendizado destes frameworks fica bem menor.

A parte de Banco de Dados foi feito em memória, gravando os objetos em memória, obviamente ao reiniciar o TomCat, os pedidos e alterações de preço feitas, irão se perder. Para evitar essa inconveniência, a classe que representa o BD cria , dentro de um bloco estático, um cardápio com preços definidos assim que é carregada.

Abaixo segue o desenho da arquitetura:



Os navegadores enviam uma request para apenas um servlet, apenas uma entrada isso gera menos portas para se preocupar com a segurança.

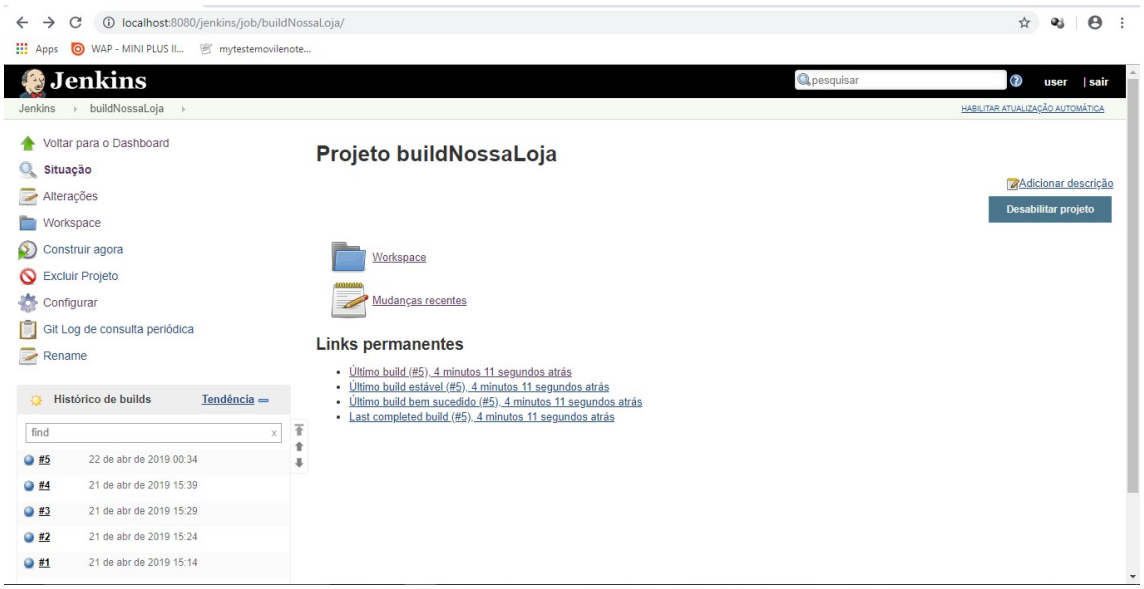
O servlet, através da interface “Acao” chama a execução da classe Java necessária, esta classe conversa com o BD se necessário e retorna uma resposta através da interface para o servlet.

Caso seja necessário o envio de uma página JSP, o servlet faz o envio da página ou se for necessário um redirecionamento no cliente-side, o servlet se encarrega de enviar.

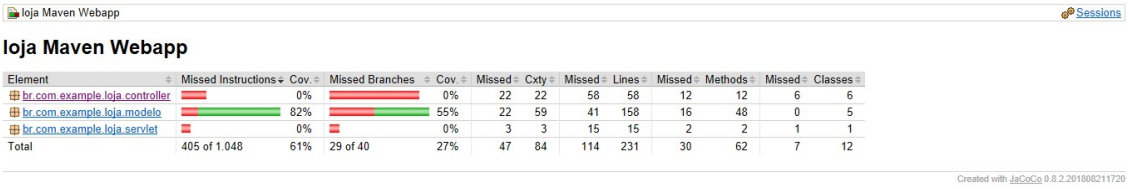
Não existe a possibilidade de acessar diretamente as páginas JSP. As páginas JSP ou o BD só são acessados via servlet, ou seja via o controller, obedecendo o padrão MVC.

A integração contínua foi feita com o Jenkins configurado como localhost tendo como target a geração de um pacote loja.war gerado pelo Maven.

A integração poderia ser melhor ainda utilizando o Jenkins na Digital Ocean, dessa forma, bastaria fazer um commit no GitHub e o Jenkins automaticamente executaria o build, mas eu precisaria de mais tempo para estudar como executar essa tarefa.



Um relatório de cobertura de testes é gerado a cada build feito. Para isso foi utilizando o plugin JACOCO que exibe de forma gráfica e bem user friendly a cobertura dos testes feitos. O relatório está na pasta Docs/Jacoco/index.html



O único item do desafio que fiquei realmente com dúvida foram os dois últimos do nível difícil. Esses eu realmente precisaria entender melhor o solicitado e estudar a melhor solução para atender.

Com relação a UI utilizei apenas o HTML, realmente com mais tempo, poderia utilizar o BootStrap e Jquery para dar uma “incrementada” no visual, mas preferi focar no backend e nos ambientes de integração contínua.