

# 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.

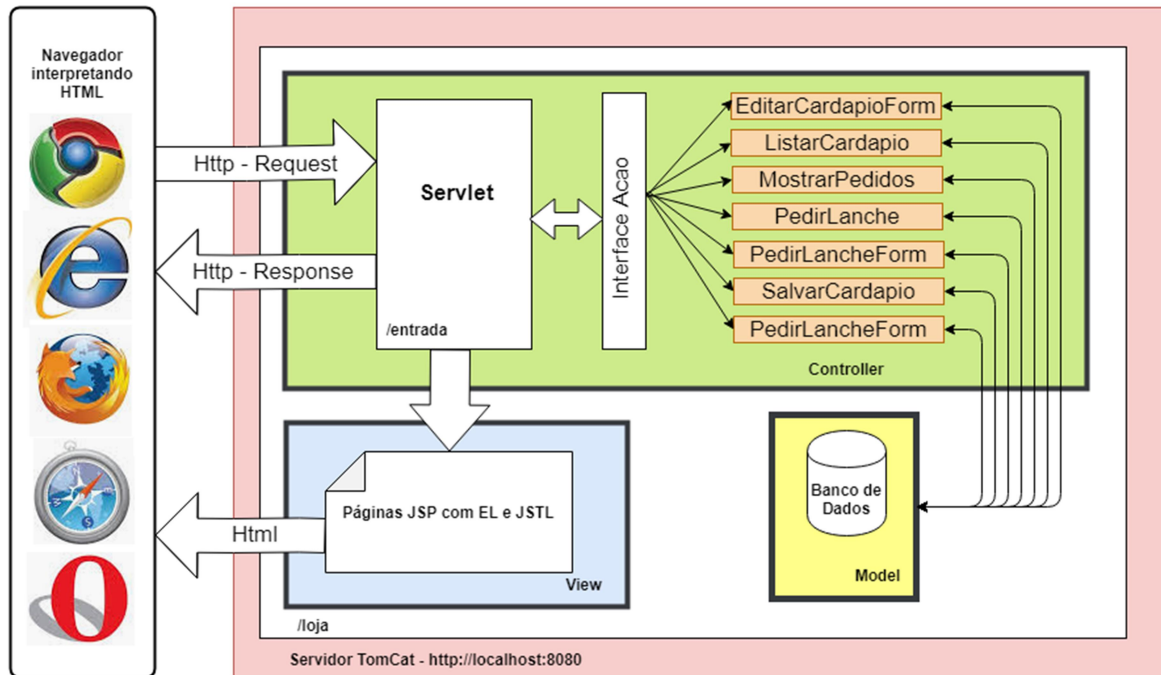
Utilizamos Expression Language (EL , `${ }`) para executar a transferência dos atributos como lanche, ingredientes etc, lá do backend para o FrontEnd.

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.

Jenkins

localhost:8080/jenkins/job/buildNossaLoja/

pesquisar user sair

HABILITAR ATUALIZAÇÃO AUTOMÁTICA

Voltar para o Dashboard

Situação

Alterações

Workspace

Construir agora

Excluir Projeto

Configurar

Git Log de consulta periódica

Rename

Projeto buildNossaLoja

Adicionar descrição

Desabilitar projeto

Workspace

Mudanças recentes

Links permanentes

- Último build (#5) 4 minutos 11 segundos atrás
- Último build estável (#5) 4 minutos 11 segundos atrás
- Último build bem sucedido (#5) 4 minutos 11 segundos atrás
- Last completed build (#5) 4 minutos 11 segundos atrás

Histórico de builds Tendência

find

#5	22 de abr de 2019 00:34
#4	21 de abr de 2019 15:39
#3	21 de abr de 2019 15:29
#2	21 de abr de 2019 15:24
#1	21 de abr de 2019 15:14

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

loja Maven Webapp Sessions

loja Maven Webapp

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
br.com.example.loja.controller		0%		0%	22	22	58	58	12	12	6	6
br.com.example.loja.modelo		82%		55%	22	59	41	158	16	48	0	5
br.com.example.loja.servlet		0%		0%	3	3	15	15	2	2	1	1
Total	405 of 1.048	61%	29 of 40	27%	47	84	114	231	30	62	7	12

Created with JaCoCo 0.8.2.201808211720