<VBW²>

<Mercado Fácil>
Documento de Arquitetura de Software

Versão <1.0>

Histórico da Revisão

Data	Versão	Descrição	Autor
<19/11/15>	<1.0>	<Aplicação da checklist de arquitetura>	<VBW²>

Índice Analítico

Documento de Arquitetura de Software

1. Introdução

1.1 Finalidade

Este documento oferece uma visão geral arquitetural abrangente do sistema, usando diversas visões arquiteturais para representar diferentes aspectos do sistema. O objetivo deste documento é capturar e comunicar as decisões arquiteturais significativas que foram tomadas em relação ao sistema Mercado Fácil.

1.2 Escopo

O documento propõe a arquitetura de software do Mercado Fácil, para futuro desenvolvimento e entrega de trabalho acadêmico.

1.3 Definições, Acrônimos e Abreviações

HTML: HiperText Markup Language

JSP: Java Server Pages

1.4 Referências

MERCADO FÁCIL. **Especificação de objetivos e requisitos**: MF-EOR-001. Goiânia, 2015.

1.5 Visão Geral

Este documento está organizado em:

Introdução: visão geral de todo o documento de arquitetura;

Representação da Arquitetura: seção que descreve qual a arquitetura e como ele é representada;

Metas e Restrições da Arquitetura: descrição dos objetivos que tem um impacto significativo na arquitetura;

Visão de Caso de Uso: onde os casos de uso ou um cenário do modelo de casos de uso são listados;

Visão Lógica: seção que descreve as partes significativas do ponto de vista da arquitetura do modelo de design, como sua divisão em subsistemas e pacotes, Além disso ela mostra sua divisão em classes e utilitários de classe.

2. Representação Arquitetural

O modelo de arquitetura de software do Mercado Fácil segue o padrão do MVC, pois o aumento da complexidade das aplicações desenvolvidas torna-se fundamental a separação destes dados, de forma que as alterações feitas no layout não afetem a manipulação de dados, e estes poderão ser organizados sem alterar o layout.

3. Metas e Restrições da Arquitetura

Algumas restrições e metas de sistema principais que possuem relação significativa à arquitetura e a implementação, são:

- Arquitetura de camadas que permita isolar as funcionalidades ligadas ao negócio das

funcionalidades relacionadas com a interface homem-computador;
- O sistemas só será acessível quando conectado a internet;
- O sistema gerenciador de banco de dados escolhido para suportar está aplicação é o MySQL.

4. Visão de Casos de Uso

Abaixo temos um lista dos casos do sistema final, demos destaque aos casos de uso em negrito, pois tem grande importância para a arquitetura e serão demonstrados posteriormente:

- **Cadastrar-se,**
- Excluir cadastro,
- **Comprar,**
- **Pesquisar por produtos,**
- Solicitar adição de anúncios,
- Selecionar forma de pagamento,
- Efetuar login,
- Solicitar adição de promoções,
- Editar dados cadastrais,
- Relatar vendas realizadas,
- Solicitar cadastro de novos produtos.

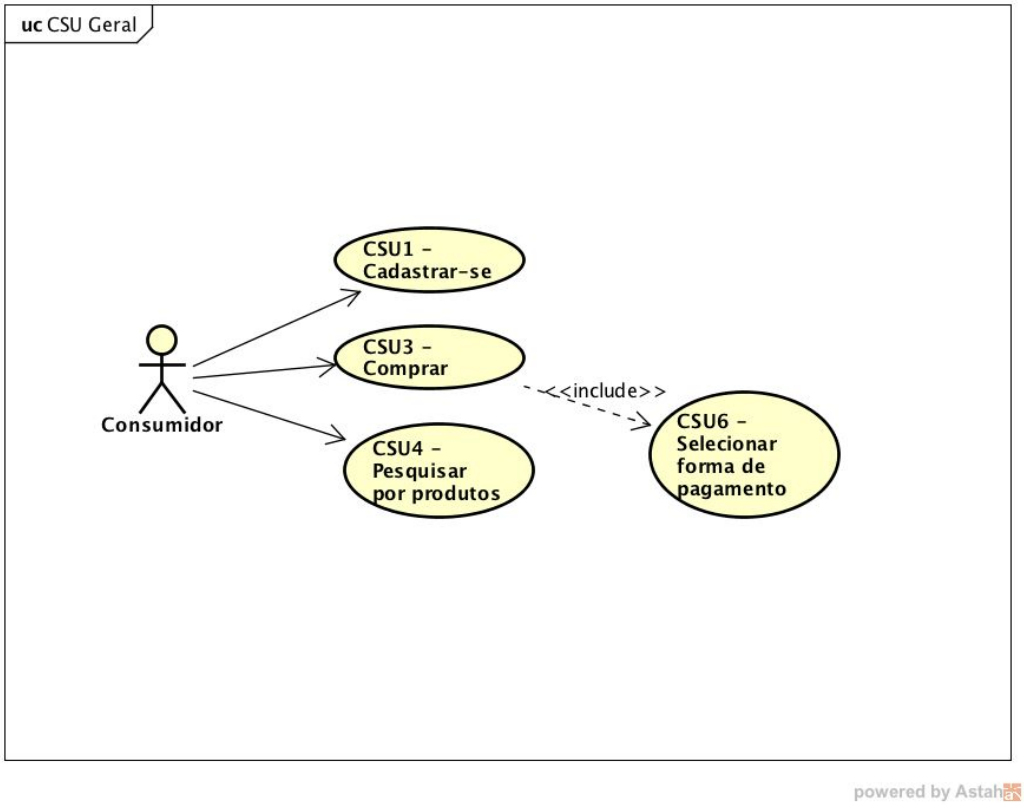
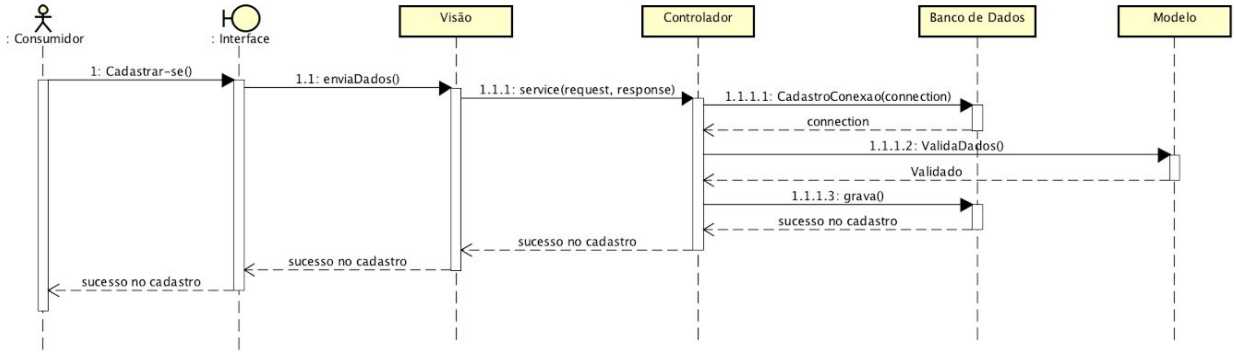
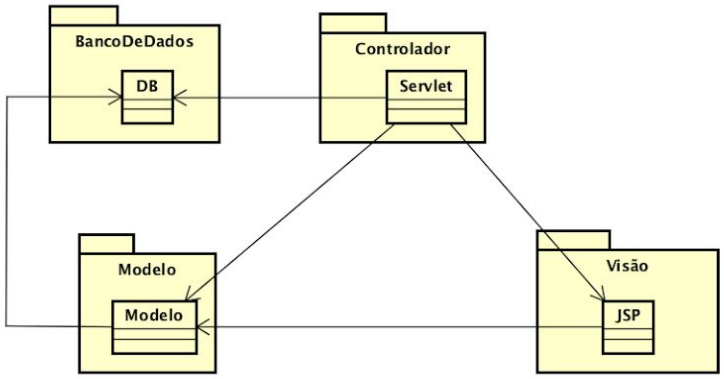


Figura 1. Diagrama geral dos casos de uso destacados

4.1 Realizações de Casos de Uso



5. Visão Lógica

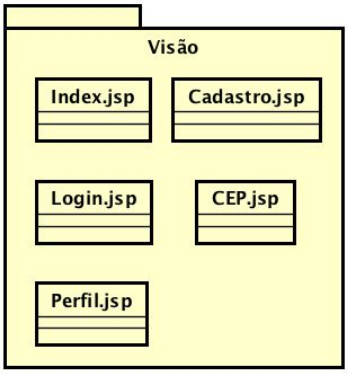


5.1 Visão Geral

O software será desenvolvido em camadas (Model-View-Controller), sendo assim, haverá separação da parte visual da parte lógica, para que haja baixo acoplamento entre as duas partes, além de facilitar a manutenibilidade e possibilitar a incrementação de novas funcionalidades que venham surgir no futuro.
Controlador: lida com as requisições dos usuários. É responsável por retornar uma resposta com a ajuda das camadas Modelo e Visão.
Modelo: responsável por obter os dados convertendo-os em conceitos significativos para sua aplicação, assim como, processar, validar, associar e qualquer outra tarefa relativa ao tratamento dos dados.
Visão: responsável por usar as informações disponibilizadas para produzir qualquer interface de apresentação que sua aplicação possa necessitar.

5.2 Pacotes de Design Significativos do Ponto de Vista da Arquitetura

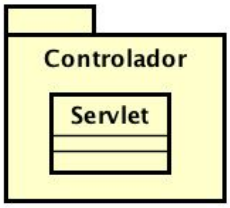
5.2.1 Visão



A visão é a camada responsável pela interface do sistema, por isso ela se torna uma das mais relevantes para a arquitetura. No pacote da visão são mostradas as principais páginas do sistema.

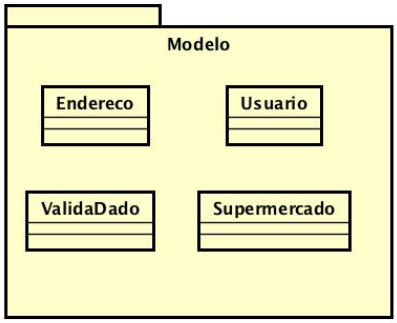
- Perfil, mostra os dados dos usuários, bem como as funcionalidades para cada um.
- Login, é página responsável pela entrada dos dados para autenticação no site.
- Cadastro, página de entrada de dados dos usuários para gravação no banco.
- Index, página principal.

5.2.2 Controlador



O controlador é o responsável por atribuir responsabilidade para outras classes ou solicitar algo. Será utilizado uma classe com nome de Servlet para fazer papel de controlador.

5.2.3 Modelo



O Modelo é responsável pelas regras de negócio, e mantém as entidades:

- Endereco, superclasse para dados de endereço.
- Usuário, superclasse para dados de usuário.
- ValidaDado, classe para validação de dados de cadastro ou login.
- Supermercado, classe que tem atributos de supermercado.

5.2.4 **Banco de Dados**

O pacote Banco de Dados tem a classe DB, responsável pelas conexões com o banco de dados.

6. **Qualidade**

A arquitetura de software suporta os requisitos de qualidade, conforme estabelecido na especificação de qualidade tratadas neste e nos outros documentos relacionado a este sistema. Assim, com estes requisitos já delimitados e detalhados temos as principais dimensões de qualidade do sistema que formam a arquitetura, em que dentre estes pontos podemos citar:

- Objetivos de portabilidade: Como já detalhando em outros documentos o software aplicativo estará disponível para todos os sistemas operacionais mobile existentes atualmente. Na versão desktop estará disponível para os navegadores mais conhecidos e mais utilizados.

- Objetivos de manutenibilidade: Por se tratar de um software construído em camadas (MVC), caso seja necessário atualizar partes do código por exemplo na camada de visão de nada irá alterar nas outras camadas, com isso a manutenibilidade se torna bem mais fácil e sem impactar outras partes

do sistema que estão funcionando corretamente.

- Objetivos de extensibilidade: A capacidade do software crescer sem causar acoplamento foi considerada, temos extensibilidade em classes dos pacotes de design, por exemplo: modelo e controlador.