**Projeto Arquitetural**

**Sobre a Organização Geral do Sistema**O sistema SrEstoque será centralizado com seus componentes  
funcionando numa única máquina cliente, e será organizado em três  
camadas logicamente interconectadas, porém funcionalmente  
distintas (ver Figura 1 abaixo).

**Figura 1**

**Camada de Apresentação (Interface com o usuário):**Forneceremos um tipo de interface cliente que será uma interface gráfica web, mais atraente e que proporcionará maior usabilidade.  
Mais adiante descreveremos quais ferramentas, quais  
componentes gráficos e quais modelos nós utilizaremos para o  
desenvolvimento desta GUI (ver seção Linguagens e Tecnologias de  
Desenvolvimento).

**Camada de Script:**Funciona como um interpretador de comandos. Ou seja, é uma  
camada que recebe comandos da UI (User Interface) sob a forma de  
string. Esta string será mapeada para uma funcionalidade fornecida  
pela camada de negócio (através da API de negócio ou API Bussiness  
Logic). Uma das vantagens do uso desta camada é que com ela  
facilmente criaremos uma forma de automação/ativação da lógica  
de negócio transparente a qualquer interface de usuário.  
Veja a tabela a seguir. Ela mostra um resumo dos scripts  
utilizados no sistema:

|  |  |
| --- | --- |
| **COMANDO** | **DESCRIÇÃO** |
| CRUD de Categoria | Gerencia de categoria |
| CRUD de Produto | Gerencia de produto |
| CRUD de Cidade | Gerencia de cidade |
| CRUD de Fornecedor | Gerencia de fornecedor |
| CRUD de Loja | Gerencia de loja |
| CRUD de Transportadora | Gerencia de transportadora |
| CRUD de Usuário | Gerencia de usuário |
| CRUD de Entrada | Gerencia de entradas |
| CRUD de Saída | Gerencia de saídas |
| CRUD de Item de Entrada | Gerencia de itens de entrada |
| CRUD de Item de Saída | Gerencia de itens de saída |
| Estoque | Controla o números de produtos em estoque fazendo a contabilidade entre itens de entrada e saída. |
| Abaixo do mínimo | Lista os produtos que estão com produtos em estoque abaixo da quantidade mínimo. |
| Por fornecedor | Lista todos os produtos do estoque separados por fornecedor |
| Por Categoria | Lista todos os produtos do estoque separados por categoria |
| Tempo médio | Informa o tempo médio esperado, em dias, levando em consideração a data de pedido e a data de entrega. |
| Saída por Loja | Informa a soma do valor total de quanto sai de mercadorias de uma loja da rede |
| Peso total | Lista o peso total gerado pelos eventos de entrada e saída de cada loja. Para |
| Frete total | Lista o valor obtido somando o frete total de cada transportadora |
| Qtd por categoria | Informa a quantidade de produtos por categoria |

**Regras de Negócios:**É a camada que contém todas as regras que faz o alocador  
funcionar corretamente. Em outras palavras, são as  
interações/colaborações entre os bussiness objects necessárias para a  
resolução das tarefas.  
As APIs são basicamente interfaces entre duas camadas, onde  
a camada superior usa os métodos fornecidos por tais interfaces para  
ter acesso as funcionalidades da camada inferior.  
**Outras Considerações**O sistema conterá uma camada para acesso a dados  
persistentes e implementará outros tipos de serviços como acesso  
multiusuário, controle de concorrência, processamento paralelo ou  
multitarefa, segurança e autenticação de  
usuário.  
  
Entre as camadas principais haverá interação frequente, porém  
com baixo volume de dados. Com relação às estruturas de  
comunicação e controle entre as camadas, optamos por utilizar o  
padrão MVC.

**Possibilidade de Reuso**Utilizando a arquitetura proposta, estaremos isolando a camada  
de regras de negócio da camada de apresentação. Isto propicia  
duas vantagens:  
•Um baixo acoplamento entre elas;  
•Um alto grau de reuso.  
Além do mais, dentro da bussiness logic, estaremos aplicando  
alguns padrões de projeto que maximizará a flexibilidade do código,  
e, portanto, facilitarão a introdução de futuros novos requisitos.  
**Linguagens e Tecnologias de Desenvolvimento**Para a implementação do sistema utilizaremos conceitos do  
paradigma orientado a objetos haja vista que os padrões de projeto  
utilizados são aplicados a este tipo de tecnologia. E a linguagem de  
programação escolhida foi à linguagem Java, já que apresenta uma  
série de vantagens como:  
•Portabilidade;  
•Grande variedade de recursos como classes utilitárias (Vector,  
HashMap, Iterator , List, ArrayList);  
•Facilidades no tratamento de exceções;  
•Familiarização dos integrantes da equipe de projeto com a  
linguagem, etc.  
Na construção do sistema, as seguintes ferramentas serão  
utilizadas:  
•Netbeans 8.1 (Um editor de programas Java distribuído livremente);  
•JDK (Java Development Kit);  
•JSF (Java Server Faces é uma especificação Java para a construção ­de interfaces de usuário baseadas em componentes para aplicações web, fornece um conjunto de componentes gráficos como  
frames,panels, lists, buttons etc, que serão utilizados para o  
desenvolvimento da interface gráfica exibidos no browser.

• Bootsrap (para o estilo das páginas web);  
• JPA (para fazer o mapeamento e persistências das entidades);

•EJB (fornecer um desenvolvimento rápido e simplificado de aplicações Java, com base em componentes distribuídos, transacionais, seguros e portáveis.)  
•stah (Para construção dos diagramas UML);  
•Maven (Para controle de dependências);

**Sobre Requisitos de Desempenho**Como qualquer sistema, nosso sistema tem que atender  
requisitos de desempenho. Porém, este não é um ponto crucial.  
Espera-se um nível mínimo de velocidade e pouco uso de recursos  
como memória e tempo de CPU.  
**O Que Deve Ser Produzido**O sistema, como já foi dito, será implementado em Java. Logo,  
na máquina cliente deverá existir uma JRE (Java Runtime Enviroment)  
para que os .class gerados sejam interpretados. Além disso deve estar instalado o container Glassfish onde será implantada a aplicação.

Também será produzido um arquivo pom.xml que é interpretado pelo maven , e prove as seguintes ações:  
•compila: compila o programa;  
•limpa: limpa a árvore do projeto de tudo que seja  
 desnecessário, incluindo arquivos .class, etc;  
•testa: execute todos os teste de unidade em interface gráfica  
• executa: para executar o programa com interface gráfica no browser.  
Quanto ao processo de instalação, basta seguir o manual de implantação que vai anexo ao sistema.  
A distribuição do software será feita livremente, sem algum custo  
financeiro ou restrições de tempo.  
**Sobre a Integração Futura**Da forma como nossa arquitetura foi sugerida, caso haja a  
necessidade futura de criação de novas camadas de apresentação  
ou até mesmo de integração com sistemas legados, nossa tarefa de  
manutenção será reduzida sensivelmente, devido a reusabilidade  
e portabilidade proposta por tal.  
**Sobre Entidades Externas ao Sistema**Neste sistema não haverá acesso externo de/a outros sistemas.