UNIVERSIDADE FEDERAL DO PIAUÍ
CAMPUS SENADOR HELVÍDIO NUNES DE BARROS
CURSO BACHARELADO EM SISTEMAS DE INFORMAÇÃO

# SISTEMA WEB PARA CONTROLE DE ESTOQUE DE UMA CENTRAL DE COOPERATIVAS - CONTESCOOP

FELIPE LUAN DE SOUSA

PICOS - PI
2015

FELIPE LUAN DE SOUSA

# SISTEMA WEB PARA CONTROLE DE ESTOQUE PARA UMA CENTRAL DE COOPERATIVAS - CONTESCOOP

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Sistemas de Informação do Campus Senador Helvídio Nunes de Barros da Universidade Federal do Piauí como parte dos requisitos para obtenção do Grau de Bacharel em Sistemas de Informação.
Orientador: Prof. Esp. Leonardo Pereira de Sousa.

PICOS - PI

2015

**FICHA CATALOGRÁFICA**

**Serviço de Processamento Técnico da Universidade Federal do Piauí**

**Biblioteca José Albano de Macêdo**

**S725s** Sousa, Felipe Luan de.

Sistema web para controle de estoque de uma central de cooperativas-contescoop / Felipe Luan de Sousa . – 2015.

CD-ROM : il.; 4 ¾ pol. (48 f.)

Monografia(Bacharelado em Sistemas de Informação) – Universidade Federal do Piauí, Picos, 2015.

Orientador(A): Prof°. Esp. Leonardo Pereira de Sousa

1. Controle de Estoque-Sistema Web. 2. Sistemas de Informações Gerenciais. 3. Sistemas-Classificação. I. Título.

**CDD 005**

SISTEMA *WEB* PARA CONTROLE DE ESTOQUE DE UMA CENTRAL DE COOPERATIVAS

FELIPE LUAN DE SOUSA

Monografia aprovada como exigência parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Data de Aprovação

Picos – PI. 30 de junho de 2015

Leonardo Pereira de Sousa
Prof. Esp. Leonardo Pereira de Sousa
Orientador

Alcilene Dalília de Sousa
Profª Me. Alcilene Dalília de Sousa
Membro

Fredison Muniz de Sousa
Prof. Esp. Fredison Muniz de Sousa
Membro

Dedico este trabalho a minha família, em especial a minha avó Maria Raimunda (in memoriam), minha mãe Maria José, minha tia Francisca Maria, meu tio Vicente Rufino e minha sobrinha Ana Carolina, por sempre acreditarem no meu potencial e me apoiarem durante todo o tempo.

## AGRADECIMENTOS

Primeiramente a Deus, pela oportunidade e privilégio que nos foram concedidos ao participar e compartilhar tamanha experiência de frequentar o curso e atentarmos para importância dos temas vivenciados diariamente ou não, mas que possuem grande importância nas nossas vidas.

A minha família pelo incentivo, carinho e confiança. Em especial a minha mãe e minha avó (*in memoriam*), pelos ensinamentos, conselhos e pelo esforço em sempre proporcionar a seus filhos tudo de melhor. A minha tia e seu esposo, pela paciência e apoio, que me ajudaram na busca e realização desse trabalho.

Aos meus irmãos, sobrinhos e sobrinhas, que estiveram sempre ao meu lado com apoio e ações, demonstrando união, que sem dúvida foi de suma importância para desenvolvimento pessoal, social e profissional.

Ao meu orientador, Leonardo Sousa, por apoiar e dividir seu conhecimento com todos(as). Agradeço pela paciência e dedicação, que foram cruciais para que a realização desse trabalho fosse alcançada.

A todos os professores do curso de Sistemas de Informação, que mesmo não sendo meus orientadores, foram co-orientadores durante todo o curso, dividindo cada gota de conhecimento que possuem.

Aos colegas e mais ainda aos amigos, que estiveram presentes em tantos momentos, tristes complicados e alegres, mas que juntos, me apoiaram e incentivaram, e até mesmo longe, com as mensagens. Agradeço por me tornarem uma pessoa melhor, na vida acadêmica, social e profissional.

A todos vocês muito obrigado!

“Por aprendizagem significativa, entendo, aquilo que provoca profunda modificação no indivíduo. Ela é penetrante, e não se limita a um aumento de conhecimento, mas abrange todas as parcelas de sua existência.”

Carl Rogers

## RESUMO

Utilizar um sistema de informação tornou-se fator decisivo e de grande importância nas empresas, visto que ele propicia o levantamento de dados e resultados precisos, além de fortalecer a atuação das empresas, gerando informações rápidas, precisas e seguras. O objetivo desse trabalho é apresentar um sistema *web* que controle o estoque de uma central de cooperativas, que proporcionará segurança e confiabilidade exigida para que, os dados sejam acessados pela central a qualquer momento, sendo incluídos os dados das cooperativas singulares. Esses dados, após devido processo de análise e refinação, tornaram-se informações, que o sistema deverá apresentar ao fim de cada processo realizado. Para alcance do objetivo, foram utilizadas no desenvolvimento metodologias como *Scrum*, modelagem UML, linguagem de programação *Ruby*, *Framework Ruby on Rails*, *Framework* de *front-end Bootstrap* e sistema de gerenciamento de banco de dados *MySql*. Por fim, foram mostradas as principais funcionalidades garantindo a conformidade com as necessidades do problema exposto.



**Palavras-chave:** Sistema *Web*, *Ruby on Rails*, Controle de Estoque.

## ABSTRACT

Using an information system has become a decisive factor of great importance in business, since it provides the data collection and accurate results, and strengthen the performance of companies, creating fast, accurate and reliable information. The aim of this paper is to present a web system that controls the stock of a central cooperatives, which provide security and reliability required for that data to be accessed by the center at any time, and included data from individual cooperatives. These data, after due process of analyzing and refining, become information, the system will present the end of each process performed. To reach the goal, were used in the development methodologies such as Scrum, UML modeling, Ruby programming language, Ruby on Rails Framework, Framework of Bootstrap front-end and database management system MySQL. Finally, we show the main features ensuring compliance with the requirements of the above problem.



**Keywords:** Web System, Ruby on Rails, Control of stock.

## LISTA DE ILUSTRAÇÕES

## LISTA DE QUADROS

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| ANSI | *American National Standards Institute* |
| COCAJUPI | Central de Cooperativas dos Cajucultores do Estado do Piauí |
| CONTESCOOP | Controle de Estoque de Cooperativas |
| CSS3 | *Cascading Style Sheets* versão 3 |
| DRY | *Don't Repeat Yourself* |
| HTML5 | *HiperText Markup Language* versão 5 |
| IRB | *Interactive Ruby* |
| MIT | *Massachusetts Institute of Technology* |
| MVC | *Model-View-Controller* |
| SAD | Sistema de Apoio a Decisão |
| SAE | Sistema de Apoio ao Executivo |
| SGBD | Sistema de Gerenciamento de Banco de Dados |
| SIG | Sistema de Informação Gerencial |
| SPT | Sistema de Processamento de Transações |
| SQL | *Structured Query Language* |
| UML | *Unified Modeling Language* |
| URL | *Uniform Resource Locator* |

**SUMÁRIO**

# 1 INTRODUÇÃO

A realização de processos que forneçam informações precisas em tempo hábil é sem dúvida uma das maiores necessidades de qualquer entidade que busque desenvolvimento e ganho de espaço no mercado atual. A informação é um recurso efetivo e inexorável para as organizações, principalmente quando planejada e disponibilizada de forma personalizada, com qualidade inquestionável e preferencialmente antecipada para facilitar as divisões (REZENDE, 2007).

Uma das maneiras mais eficientes que fora encontrada ao longo da história e que hoje englobam boa parte de todos os processos internos e externos dessas entidades, são os sistemas de informação. Segundo Tonsig (2000), “sistema é um conjunto de entidades relacionadas, que interagem buscando atingir um objetivo declarado e outros correlatos”.

Sistema de Informação é um conjunto de elementos como: pessoas, computadores, processos, redes de computadores, *software*, *hardware*, entre outros que tem por função organizar e disseminar informações (O’BRIEN, 2006). Nesse contexto, utilização desses meios é de fundamental importância, para que as organizações e, sobretudo as pessoas consigam atingir os objetivos desejados, levando em consideração o tempo e qualidade.

Sejam em organizações públicas ou privadas, a utilização dos sistemas de informação têm sido implantados em todos os setores, devolvendo resultados de processos que vão desde o nível operacional até o estratégico, trazendo como resultado uma série de ganhos, como por exemplo, a segurança das informações obtidas e o rápido acesso a estas.

A Central de Cooperativas dos Cajucultores do Estado do Piauí (COCAJUPI) é formada pela junção de 09(nove) cooperativas, para beneficiamento e escoamento da produção dos sócios das singulares(cooperativas filiadas a central). Segundo Carvalho (2011), “o surgimento dos meios cooperativos atende as necessidades que visam estabelecer o direcionamento necessário para a busca de objetivos.”.

Buscando o desenvolvimento socioeconômico dos sócios e com a necessidade básica de qualquer empresa, instituição comercial ou qualquer coisa do gênero, a COCAJUPI e suas singulares precisam encontrar formas adequadas para controlar o estoque de seus produtos, analisando desde o processo de entrada e suas respectivas saídas. Utilizando planilhas eletrônicas que foram desenvolvidas ao longo dos anos, hoje a instituição obtém resultados até certo ponto satisfatórios, porém, conseguidos de forma trabalhosa, onde, a obtenção dos

mesmos, é feita manualmente, tanto nas entradas e mais trabalhosa ainda para realização das saídas, visto que o levantamento das informações necessárias devem ser precisas e seguras.

Tendo em vista a necessidade descrita, bem como a solução utilizada até o presente momento, nota-se a deficiência em adquirir essas informações de forma mais prática, robusta e segura, sem que o trabalho manual seja desgastante e repetitivo.

## 1.1 Objetivo

O objetivo desse trabalho é desenvolver um sistema *Web* para controle de estoque da Central de Cooperativa dos Cajucultores do Estado do Piauí – COCAJUPI.

## 1.2 Organização do Documento

Após a introdução serão apresentados capítulos que estão organizados da seguinte maneira:

- Capítulo 2 – Referencial Teórico: Fornece o embasamento teórico para o trabalho. São demostrados conceitos relacionados a tecnologias e metodologias utilizadas no desenvolvimento do referido sistema.
- Capítulo 3 – Sistema *Web* para controle de estoque de uma central de cooperativas: Serão mostradas as etapas de análise de requisitos, os diagramas produzidos, que foram de suma importância para o entendimento do problema em questão e o desenvolvimento do sistema.
- Capítulo 4 – Funcionamento do Sistema: São mostradas algumas das funcionalidades que cada usuário poderá realizar na execução do sistema.
- Capítulo 5 – Considerações Finais: Apresenta-se a conclusão do protótipo e indicações para trabalhos futuros.

# 2 REFERENCIAL TEÓRICO

## 2.1 Sistemas de Informação

Segundo O'Brien (2006), sistema pode ser definido como um conjunto de elementos inter-relacionados que operam em rumo a uma meta comum, recebendo insumos e produzindo resultados em um processo organizado de transformação. Assim, podemos identificar uma diversidade de sistemas, exercidos e executados no dia-a-dia, levando em consideração não somente os meios e metodologias identificadas para tal, mas o resultado satisfatório para necessidade ou problema em questão.

Sistema possui três componentes básicos facilmente notados: entrada (ocorre a inserção de dados referentes ao meio onde se encontra o sistema), processamento (análise e refinamento das entradas) e saída (resultado da análise e refinamento). Os sistemas podem ser considerados: subsistemas (quando fazem parte de outro sistema maior), abertos (quando estão conectados a outros por meio de *interfaces*) e adaptável (tem a capacidade de adaptar-se ou ao ambiente que está inserido). Rezende (2007), afirma que:

> "Os sistemas procuram atuar como ferramentas para exercer o funcionamento complexo das organizações, possibilitando uma avaliação analítica e até mesmo sistemática, facilitando processos internos e externos com suas respectivas intensidade e relações."

Sistemas de informação fazem parte de uma organização maior e estão em constante troca de informações entre as partes que compõem o sistema ao qual estão inseridos. Rezende (2007), afirma que "Todo sistema, usando ou não recursos de tecnologia da informação, que manipulam dados e gera informação pode ser genericamente considerado sistema de informação.".

Segundo Laudon e Laudon (2004) "Sistema de informação pode ser definido tecnicamente como um conjunto de componentes inter-relacionados que coleta (recupera), processa, armazena, e distribui informações destinadas a apoiar a tomada de decisões, a coordenação e o controle de uma organização.".

Coletar o máximo de dados sobre determinado problema; analisar e processar esses e por fim, distribuir informações que ajudem a melhorar o desempenho da organização é o

papel principal de um sistema de informação, que são implementados e implantados visando atender essa necessidade. Levando em conta não apenas os resultados desejados, mas também a forma como são encontrados, a segurança e a satisfação das pessoas que se encontram envolvidas direta ou indiretamente com esses, são também fatores de suma importância para escolha e utilização de um sistema de informação. De acordo O'Brien (2006):

> "Sistema de informação é um conjunto organizado de pessoas, *hardware*, *software*, redes de comunicações e recursos de dados que coleta, transforma e dissemina informações em uma organização.".

## 2.2 Classificação dos Sistemas

Segundo Laudon e Laudon(2004), nenhum sistema isolado consegue fornecer todas as informações necessárias a uma organização. Justamente por essa afirmação e por também existirem sempre diferentes pontos de vista ou mesmo interesses dependendo de que nível estratégico ou organizacional, é que existem diversos tipos de sistemas.

Diante do vasto conglomerado de organizações e suas metodologias diferentes, (XEXÉO, 2007) afirma, que os sistemas de informação atualmente servem em todas as áreas e níveis das organizações, sendo considerados como ferramenta essencial para o sucesso de suas atividades. De acordo com a afirmação do autor, podemos classificar esses sistemas em quatro tipos principais:

- Sistemas de processamento de transações (SPTs): atendem ao nível operacional da organização, realizando e registrando transações rotineiras e atividades básicas da organização, por isso considerado muito importante, visto que para seu funcionamento por poucas horas, podem causar grandes danos a organização e demais interessados;
- Sistemas de informação gerencial (SIG): inseridos no nível gerencial da organização, resumem e relatam as operações básicas da organização. Não são flexíveis e possuem sua capacidade analítica reduzida, sendo que em geral, respondem a questões especificadas anteriormente, apoiando as funções de planejamento, controle e decisão no nível em que se encontram;
- Sistemas de apoio a decisão (SAD): atendem ao nível de gerencia da organização, ajudando os gerentes a tomarem decisões não-usuais, que se alteram com rapidez e que não possuem resolução predefinida, para isso, além de utilizar informações dos sistemas listados anteriormente, também tem acesso a fontes externas;

- Sistemas de apoio ao executivo (SAEs): atendem a gerência sênior da organização, ajudando na tomada de decisões não rotineiras e sem procedimento prévio, incorporando dados externos, bem como do SIG e SAD internos, exibindo apenas o que for mais relevante ao seu usuário;

Os sistemas descritos de acordo com seus níveis específicos tem fundamental importância, por facilitar a realização das ações a serem desenvolvidas pelos seus usuários, através destes, em qualquer lugar onde lhes seja permitido realizar seu trabalho, de forma eficiente e segura. Como é o caso dos Sistemas *web* em que seus usuários podem acessar qualquer informação independentemente de onde estejam.

O presente sistema segue os conceitos dos Sistemas de informações Gerenciais, devido a necessidade de controlar e gerir as informações do estoque da instituição.

### 2.2.1 Sistemas *Web*

O termo aplicação *web* pode ser utilizado para referir-se a qualquer *software* baseado na Internet, que realize ações e/ou funcionalidades de acordo com uma entrada dados, que é feita por usuário final ou não. Pauli (2013) explica que o termo aplicação *web* possui significados diferentes para pessoas diferentes, pois, sempre que conversamos e contextualizamos algo, as reações de percepções podem ser diferentes e em si tratando de aplicação *web*, a utilização dos termos aplicação *web*, *web site*, sistema baseado em *web*, *software* baseado em *web* ou simplesmente *web*, todos eles poderão ter o mesmo significado.

Segundo conceitos mais técnicos *Web* é apenas um dos serviços disponibilizados pela internet, onde uns conjuntos de páginas estão acessíveis a todos e em qualquer lugar, através de hipertexto e *links*. (RODRIGUES, 2010) Descreve a *web* como um sistema de informação em hipertexto, gráfico, distribuído, independente de plataforma, dinâmico, interativo e global, utilizado na internet.

Silva (2008) afirma que não existe uma definição exata que diferencie claramente *sites* e aplicações *web.* Alguns afirmam que *sites* são páginas da internet que possuem somente leitura, onde seus usuários podem apenas visualizar informações, sem que haja interação entre ambas as partes. Enquanto isso, as aplicações *web* são páginas da internet que possibilitam a leitura e escrita, ou seja, existe a interação do usuário com o sistema, podendo ocorrer tanto recepção quanto inserção de dados, estabelecendo assim, a comunicação entre usuários e sistemas.

Assim, podemos afirmar que, os sistemas *web* auxiliam seus usuários a executarem determinadas tarefas e até mesmo o gerenciamento de dados e informações, ajudando a organização tanto nos processos rotineiros quanto nas tomadas de decisões internas e externas, estando seus usuários ou não na sede da organização.

### 2.2.2 Sistemas de Informações Gerenciais

Gordon e Gordon (2006) definem dados como fatos, valores, observações e medidas que não estão sendo contextualizadas ou organizadas de forma que uma pessoa possa compreender e utilizar. Entretanto, informação, nada mais é que os dados devidamente processados, ou seja, organizados, analisados e interpretados para que fosse possível obter seu real significado. Tendo em vista o que foi afirmado anteriormente, podemos enfatizar que os dados ao serem processados, analisados e organizados, geram informações, que possuem significados. Os autores indicam que os gestores das organizações, podem utilizar as informações para obter conhecimento, ou seja, o conhecimento proporciona uma estrutura para interpretação das informações.

Segundo Gordon e Gordon(2006) as tecnologias de informação facilitam as comunicações entre pessoas dentro das organizações, além de permitir a gestão eficaz e eficientemente de suas informações. Os sistemas com base na Tecnologia da informação, tem capacidade de otimizar as organizações, visto que, melhoram a busca por resultados mais precisos, além de facilitar todo o processo.

Cruz (2009) diz que Sistemas de Informações Gerenciais ou SIG’s, são um conjunto de programas desenvolvidos para operar e/ou administrar qualquer organização. É um dos principais responsáveis por coletar e processar dados, resultando em informações e conhecimento para quem o utiliza.

Laudon e Laudon (2004) afirmam que os Sistemas de Informação Gerencial, possuem um foco maior em eventos internos da organização apoiando funções de planejamento, controle e decisão no nível gerencial.

De acordo com as afirmações da seção 2.2.2, podemos constatar que utilizar SIG’s nas organizações, ajuda na busca por melhores resultados, otimizando o processo de levantamento, análise e organização de dados em informações, podendo, sobretudo, auxiliar na execução das diversas tarefas, agilizando e melhorando o tempo de execução, garantindo mais segurança nas tomadas de decisões, tendo em vista, que as informações são mais bem trabalhadas e alcançadas rapidamente.

## 2.3 Tecnologias

Para desenvolvimento de um projeto, faz-se necessário a utilização de recursos importantes, visando a otimização do ciclo de desenvolvimento e a obtenção de bons resultados. Em si tratando do desenvolvimento de sistemas sejam *web* ou *desktop,* existem tecnologias que são comuns a ambos, a citar linguagens de programação e banco de dados, que podem ser utilizadas para atingir o objetivo. Nesta seção, abordaremos as principais tecnologias utilizadas durante o desenvolvimento desse projeto.

### 2.3.1 Linguagem *Ruby*

A comunicação entre homem e computador, se dá por meio de *softwares* implementados com os artifícios que chamamos de linguagem de programação. Melo e Silva (2003), diz que linguagem de programação é um conjunto de recursos que podem ser compostos para construir sistemas específicos, ou seja, são códigos utilizados por profissionais da área da computação para realização e desenvolvimento de atividades das mais diversas formas.

Segundo Velloso (2003) é através do *software*, o homem se comunica com o computador, em que a linguagem de programação é a base responsável pela construção de um *software*. O autor afirma ainda, que Linguagem de Programação é um conjunto de termos (vocabulário) e de regras (sintaxe) que permitem a formulação de instruções a um computador.

De acordo com Velloso (2003), hoje, existem diversas Linguagens de Programação, onde cada uma é voltada para determinada finalidade e mais atraente para cada desenvolvedor seguindo suas características de escolha e manipulação.

*Ruby* é uma linguagem de programação criada por Yukihiro "Matz" Matsumoto, em 1995 no Japão, orientada a objeto, com tipagem dinâmica e forte, gerenciamento de memória automático. Matz queria uma linguagem de *script* que fosse mais poderosa do que Perl, e ainda mais orientada a objetos do que o *Python*.

As características que tornam a linguagem simples, flexível e que facilitou a expansão de sua utilização pelo mundo de forma rápida, foram: possuir paradigmas de implementação funcional, reflexivo, orientado a objetos e imperativo, considerado assim multiparadigma; sintaxe simples e multiplataforma, podendo ser executado em diversos

sistemas operacionais. Oliveira (2006) relaciona alguns recursos disponíveis na linguagem, que sem dúvidas que chama a atenção dos mais diversos desenvolvedores, desde leigos até os mais estáveis:

- Possui sintaxe simples, parcialmente inspiradas por Eiffel e Ada;
- Possui recursos para tratamento de exceções, assim como na linguagem *Java* e *Python*, facilitando o tratamento de erros;
- Operadores podem ser redefinidos facilmente por métodos;
- Orientada a objetos, isso significa que todo dado em *Ruby* é um objeto;
- Desenvolvido para ser aberto a melhorias. Com isso é possível que uma instância de uma classe possa se comportar diferentemente de outras instancias da mesma classe;
- Possui herança única, com a possibilidade de estender módulos. Estes módulos são coleções de métodos, sendo assim, toda classe pode importar um modulo e pegar todos os métodos;
- Possui blocos em sua sintaxe que podem ser passados para os métodos;
- Com a utilização do *garbagecoletor*, que atua em todos os objetos, não há necessidade de manter uma contagem de referências em bibliotecas externas;
- Inteiros são usados sem contar sua representação interna. Há inteiros pequenos e grandes, porém não e preciso definir qual será utilizado devido à ocorrência automática de conversão;
- Se o sistema operacional permitir, há a possibilidade de carregar bibliotecas de extensão dinamicamente;
- Suporte a *threads*;
- Possui grande portabilidade entre os sistemas operacionais.

Foi escrito na linguagem de programação C, é uma linguagem interpretada e não compilada, ou seja, para execução de um código *Ruby* é necessária a instalação de um interpretador. Segundo Flanagan e Matsumoto(2008), o interpretador pode possuir a implementação de referência que define a linguagem MRI (“Mat’z *Ruby implementation*”) que é baseada em C, mas há implementações alternativas como *JRuby* (baseada em Java), *IronRuby*( baseada em .NET), *Rubinius* (baseada em *Smalltalk*), entre outras.

Para programadores que costumam desenvolver utilizando testes, o *Ruby* dispõem do IRB (*Interactive Ruby Shell*) que é um dos principais recursos disponíveis na linguagem. Funciona como um console/terminal, em que os comandos são interpretados ao mesmo tempo

em que vão sendo inseridos. O IRB avalia cada linha inserida e já mostra o resultado imediatamente (CAELUM, 2014). Além disso, *Ruby* apresenta alguns recursos interessantes que foram inspirados e aproveitados de outras linguagens de sucesso, tornando-a assim uma linguagem bastante flexível.

Souza (2012) afirma que a criação do *Rails* foi o que fez a linguagem se tornar um sucesso. A partir da criação do *Rails*, o mercado de crescimento da linguagem tem aumentado de forma espetacular, sendo considerada hoje uma das linguagens mais utilizadas E que existe um mercado grande em torno da linguagem e a previsão é que ele continue crescendo.

### 2.3.2 *Gems*

Assim como nas diversas linguagens de programação modernas, o *Ruby* possui a utilização de bibliotecas que auxiliam e aceleram o desenvolvimento de projetos. No universo do *Ruby*, essas bibliotecas são encontradas através de *Gems*, que são arquivos com extensão .gem, onde localiza-se o código fonte da biblioteca e todas as informações e metadados, a exemplo nome e versão.

Para utilização de uma *Gem*que não esteja nos códigos fontes padrão do Ruby, é necessário abrir o arquivo *GemFile* e especificar a gem e a versão requerida. Em seguida rodar o comando *bundleinstall* no terminal e o download dos pacotes necessários são utilizados.

No projeto, fizemos a utilização *Gem "bootstrap-sass ~>3.3.4"*, versão mais atualizada. É uma versão sass do *bootstrap* que é integrada ao *Rails* e através desse, podemos melhorar a aparência do sistema, como estrutura da view e ícones pertencentes a esta.

### 2.3.3 *Framework Ruby on Rails*

*RubyonRails* é um *framework* de desenvolvimento *web* que facilita a criação, implantação e manutenção de aplicações *web*. Criado em 2003 por David HeinemeierHansson, é um *framework* de que inclui tudo que é necessário para criar e implantar aplicações *Web*. É escrito na linguagem *Ruby*, utilizando paradigma de orientação a objeto.

*RubyonRails* é um *framework* que possui código aberto, disponível sob a licença MIT permissiva; possui um *design* compacto e elegante; através da exploração da maleabilidade da linguagem *Ruby*, o *Rails* cria uma linguagem de domínio específico para escrever aplicações *web*.

Ele é desenvolvido sobre a arquitetura MVC (*Model-View-Controller)*, que separa a camada de visualização das camadas de lógicas e regras de negócio da aplicação. Essa separação é aconselhada e muito importante, porque as aplicações costumam ser desenvolvidas em navegadores específicos, mas na prática de utilização, há uma grande variedade de navegadores que são utilizados, com padrões e estilos de páginas diferentes. Com isso, a separação da camada de visão permite que o problema de incompatibilidade dos navegadores seja tratado em uma camada apenas, e não na aplicação com um todo (BURBECK, 1992). Segundo Fuentes(2012) essa arquitetura MVC – *Model-view-controller*, ou seja, Modelo, apresentação e controle, pode ser definida da seguinte maneira:

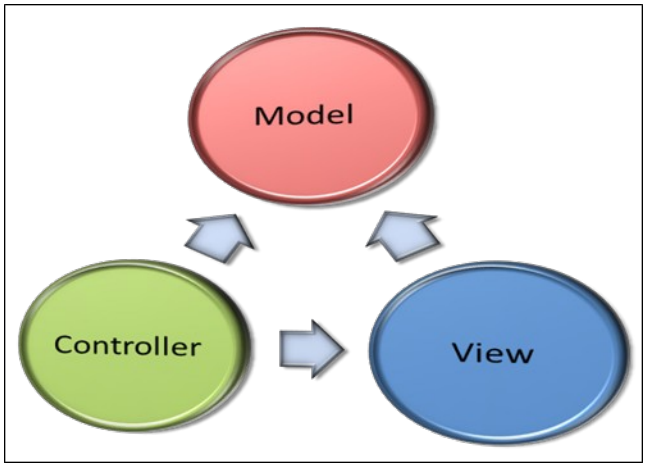



Figura 01 – Estrutura MVC

Fonte: Alistair Doulin (2015)

- Modelo possui duas responsabilidades: eles são os dados que normalmente ficam persistidos em um ou mais banco de dados (seu perfil de usuário, por exemplo). Eles também fazem parte da regra de negócio, ou seja, cálculos e outros procedimentos, como verificar se uma senha é válida;
- Controle é a camada intermediária entre a *Web* e o seu sistema. Ele pega os dados que vem de parâmetros na URL e/ou de um formulário e repassa para os modelos, que vão fazer o trabalho pesado. Em seguida, pega o resultado e transforma da maneira adequada para a Apresentação;
- Apresentação é como o aplicativo mostra o resultado das operações e os dados. Normalmente podem ser uma bela página usando as novas tecnologias de CSS3

(*Cascading Style Sheets*) e HTML5 (*Hypertext Markup Language*, versão 5) a até pequenas representações de objetos em JSON (*Java Script Object Notation*).

*Ruby on Rails* é um *metaframework*, pois se baseia em vários padrões e por ser composto de *frameworks*. Entre os *frameworks* pode-se citar como exemplo (CAELUM, 2014), *Active Record*: que funciona implementando o acesso ao banco de dados de forma transparente ao usuário e fazendo o mapeamento de estruturas relacionais. Uma das melhores facilidades que esse *framework* proporciona é o *Active Model* que inclui diversas ferramentas para validação de atributos, integração de formulários, entre outras.

O *Rails* possui três princípios básicos, o primeiro é o DRY – *dont repeat yourself* (não se repita) que estimula a diminuição do trabalho, otimizando a produtividade, assim, determinada característica já desenvolvida não precisa ser redeclarada em outras partes do projeto, bastando apenas ser chamado e adaptado para onde for necessária sua utilização. O segundo é o de Convenções sobre configurações, que quando seguidas evitam que o programador escreva códigos desnecessários; e por fim, tem-se a Automação de tarefas, que incentiva o programador a buscar resolver problemas interessantes, deixando de lado tarefas repetitivas que venha surgindo durante o projeto.

### 2.3.4 *Framework* de *Front-end Bootstrap*

Na padronização das *views*, foi utilizado o *framework Bootstrap*, buscando melhorar a apresentação do sistema, pois o padrão que o *Ruby on Rails* oferece é muito básico, sem efeitos mais significativos e atraentes para um sistema em funcionamento. Desenvolvido por Mark Otto e Jacob Thornton, é um *framework* de código aberto que padroniza o *front-end.*

Utilizando ele, o desenvolvedor fica disposto a um conjunto de vários elementos de interface, como botões já padronizados, mas que permite a alteração de suas configurações. Além disso, o mesmo dispõe de ferramentas com as convenções padrão de classes com documentação prática para melhorar os códigos que estão prontos para serem usados e personalizados de acordo com as necessidades dos desenvolvedores. Não é uma solução mágica para resolver o problema de reutilização de elementos da *interface*, mas é um bom início (MAGNO, 2013).

#### 2.3.4.1 *Sublime text 2*

Sublime Text, é um editor de códigos que possui suporte para as mais variadas linguagens existentes. Possui *design* elegante, fato que juntamente com a sua gratuidade atraiu uma grande quantidade de usuários adeptos em todo o mundo. Possui grande compatibilidade com sistemas *web* permitindo gerenciar todas as linguagens em um mesmo editor, ou seja, não sendo necessária a utilização de outros editores que trabalhem apenas com determinada linguagem.

### 2.3.5 *MySql*

É um sistema de gerenciamento de banco de dados (SGBD), que utiliza a linguagem SQL (Linguagem de Consulta Estruturada, do inglês *Structured Query Language*) como interface. É atualmente um dos bancos de dados mais populares do mundo, com mais de 10 milhões de instalações. Foi criado na Suécia por dois suecos e um finlandês: David Axmark, Allan Larsson e Michael "Monty" Widenius, que têm trabalhado juntos desde a década de 1980.

Foi desenvolvido reunindo as características como robustez, segurança e confiabilidade, já existentes nos SGBDs dominantes na época, tendo como incremento o armazenamento e gerenciamento de dados mais rápido, flexível e seguro.

A partir da quinta versão, o MSQL deixou de ser utilizado apenas na internet, para ser utilizado em aplicações locais, pois passou a ter características compatíveis com os servidores de banco de dados que tinha capacidade mais avançada e que davam suporte as reais necessidades do SGBD.

Segundo Muto (2006, p. 184), “suporta uma dúzia de tipos de dados, funções SQL, através do ANSI SQL”. Isso acaba por dar maior credibilidade e aceitação pelos usuários, além de propiciar o desenvolvimento de boas aplicações. Lobo (2008, p.07) diz:

> “Um banco de dados é um conjunto de dados armazenado em um computador. Esses dados, se observados separadamente, não têm valor nenhum, mas, quando utilizados em ordem, revelam informações que poderão ser usadas futuramente, por isso, um banco de dados deve ser seguro e nunca ficar exposto a pessoas não autorizadas.”

O Sistema de Gerenciamento de Banco de Dados (SGBD) é a ferramenta indispensável, quando se trata de gerenciar informações no banco de dados. "É um *software* projetado para auxiliar a manutenção e utilização de vastos conjuntos de dados" (GEHRKE, et.al., 2011).

## 2.4 Engenharia de Software

De acordo com Pressman(2011), "engenharia de *software* é um conjunto de três elementos – métodos, ferramentas e procedimentos, que possibilitam o controle do processo de desenvolvimento de software e oferece a base para construção com qualidade produtiva.", ou seja, a junção dos três elementos que fazem a engenharia, buscar a melhoria na construção e desenvolvimento do projeto, garantindo que esse seja alcançado com produtividade elevada e qualidade. Deve-se levar em consideração, que esses elementos não se relacionam apenas com o desenvolvimento, mas também com todas as atividades que apoiam o desenvolvimento.

O desenvolvimento de um sistema, independente da sua complexidade, é organizado por meio de processos. Segundo Pressman (2011), processo de *software* é um conjunto de atividades, ações e tarefas para o desenvolvimento de um *software*, onde, cada uma dessas partes, está contida dentro de modelos que regem o relacionamento seja do processo ou mesmo com outras partes. Esses modelos são conhecidos por Modelos de Processo, que serão abordados sucintamente a seguir.

Os primeiros modelos que surgiram, foram os de processo prescritivos, como o nome já diz, "prescrevem", ou seja, buscam ordenar o desenvolvimento de *software*, através da estrutura e ordem, propondo um conjunto de atividades e fluxo para realizá-las (PRESSMAN, 2011). Segue alguns dos modelos prescritivos mais utilizados:

- Cascata ou paradigma de ciclo de vida: requer uma abordagem sequencial, pois segue uma sequencia de passos consecutivos e dependentes; na sistemática, possui a mesma sequencia de passos todas as vezes que é aplicado. Possui as fases de levantamento de necessidades dos usuários, planejamento das atividades, modelagem e projeto, desenvolvimento da aplicação, teste, implantação e suporte. Uma fase só pode ser iniciada após o término e aprovação da fase anterior a ela. É o modelo mais antigo e mais utilizado na engenharia de *software*;
- Espiral: é desenvolvido em uma série de iterações que passam pelos processos de planejamento, análise de riscos, engenharia (ou desenvolvimento) e avaliação do

cliente. A cada nova interação é percorrida todos os processos, possibilitando adicionar novas funcionalidades sempre que identificada as necessidades;

- Evolucionário: é também iterativo, considera a ideia que os *softwares* evoluem ao longo do desenvolvimento. Na prototipagem ou versões provisórias, segue-se o modelo espiral e ao final de cada processo há a entrega de um "produto", dai entrega evolutiva, considerando que os softwares evoluem de acordo com as mudanças nas regras de negócio até a entrega final.

Todos os modelos citados são prescritivos muito utilizados atualmente, mas que, por necessitarem de muito tempo para planejamento, projeto e documentação, oportunizou o surgimento de outras metodologias conhecidas como Ágeis. Essas focalizam o desenvolvedor no *software*, a concepção e documentação um pouco mais de atrás. Destacamos abaixo, a metodologia *Scrum*, que foi utilizada no desenvolvimento do projeto.

Embora o *Scrum* atualmente seja muito utilizado em desenvolvimento de *software*, ele foi inicialmente concebido como um estilo de gerenciamento de projetos em empresas de fabricação de automóveis e produtos de consumo, por Takeuchi e Nonaka no artigo "*The New Product Development Game"*. Eles notaram que projetos, usando equipes pequenas e multidisciplinares (cross-functional) produziram os melhores resultados e associaram estas equipes altamente eficazes a formação *Scrum* do *Ruby*. Em 1995, Ken Schwaber formalizou a definição de *Scrum* e ajudou a implantá-lo em desenvolvimento de software em todo o mundo.

Para Pressman (2011) *Scrum* é um método de desenvolvimento ágil de *software* e seus princípios estão de acordo com o manifesto ágil e são usados para orientar as atividades de desenvolvimento dentro de um processo que incorpora as seguintes atividades: requisitos, análise, projeto, evolução e entrega. É utilizado principalmente para o gerenciamento de projetos de desenvolvimento de *software*, assim como outras metodologias de desenvolvimento, no entanto, pode também ser utilizado para a gerência de equipes de manutenção ou como *Scrum* de *Scrums*, que vem a ser uma abordagem para gestão de programas.

É constituído por Time *Scrum*, ou seja, o grupo *Scrum* e seus papéis associados, artefatos e regras. Um *Time Scrum*, deve otimizar a flexibilidade e produtividade, sendo auto organizável e trabalhando em iterações. Cada *Time Scrum* é composto por três papéis básicos, o *Scrum Master*, é responsável por garantir que o processo seja entendido e seguido; o *Produt Owner,* responsável por maximizar o valor do trabalho que o grupo faz; e o *Time*, consiste de

desenvolvedores com habilidades necessárias para transformar o *Produto Owner* em um pedaço potencialmente entregável do produto ao final da *Sprint*.

Esta metodologia emprega eventos com duração fixa chamados de *Time-boxes* para criar regularidade. Como seus elementos têm duração fixa, é necessária uma Reunião de Planejamento da *Sprint*, Revisão da *Sprint*, Retrospectiva da *Sprint* e a Reunião diária. Bem como dois artefatos: o *Backlog* do Produto e o *Burndown Chart*. De acordo Sabbagh (2013) *Scrum* trará benefícios se bem utilizado e seu uso aumenta a qualidade do produto e melhora a produtividade das equipes.

### 2.4.1 Engenharia de requisitos

A engenharia de requisitos é uma das atividades da engenharia de *software* que tem início na fase de comunicação e percorre pela modelagem, visando o entendimento dos requisitos do sistema em questão, através de técnicas. Essas técnicas têm por finalidade levantar os requisitos necessários para garantir que o sistema será realmente útil para quem for utiliza-lo.

Assim, Nunes (2013) afirma que requisito num sistema é uma funcionalidade ou característica considerada relevante na óptica do utilizado, representando o comportamento esperado do sistema. Os requisitos podem ser organizados em três tipos distintos:

- Requisitos funcionais: compreendidos pelos usuários e definem as funcionalidades que o sistema deve ter, bem como as restrições necessárias no mesmo. Podem ser subdivididos em requisitos funcionais, requisitos comportamentais e requisitos estruturais;
- Requisitos não funcionais ou de qualidade: declaram características de qualidade que o sistema deve ter sendo estas relacionadas às suas funcionalidades. Geralmente definem questões como desempenho e confiabilidade, acobertando questões que os requisitos funcionais não suprem;
- Requisitos de domínio ou restrições: podem restringir o próprio do sistema e refletem características e restrições desse domínio. Não são implementados, mas obrigatoriamente devem ser cumpridos.

A engenharia de requisitos é indispensável no desenvolvimento de sistemas que buscam satisfazer seus clientes, dentro dos prazos determinados e com orçamento seguido a risca.

### 2.4.2 UML

UML – *Unified Modeling Language* (Linguagem de Modelagem Unificada) é uma linguagem de modelagem ou conjunto de notações e semântica que visa representar visualmente as perspectiva de um sistema. É uma linguagem constituída de elementos gráficos sendo utilizados na modelagem e que permitem representar de forma especial os conceitos do paradigma de orientação a objetos.

Bezerra (2007) afirma que a UML é independente das linguagens de programação e de processos de desenvolvimento, ou seja, pode ser utilizada no desenvolvimento de projetos, não sendo obrigatório para o desenvolvedor utilizar um padrão de linguagem ou metodologia de desenvolvimento específica da UML.

Segundo Guedes (2011, p.23) “Durante a análise de requisitos, uma linguagem de modelagem auxilia a levantar questões que não foram concebidas durante as entrevistas iniciais”. A UML possui vários diagramas e cada um desses apresenta o sistema de uma visão diferente. Alguns dos diagramas oferecidos pela UML são:

- Diagramas de casos de uso: possuem linguagem simples e representam as funcionalidades externas do sistema, ou seja, modelando os requisitos funcionais do mesmo. Representam também a visão do usuário e quais funcionalidades esses possuem acesso. Direciona as tarefas posteriores do ciclo de vida do sistema e força os desenvolvedores a moldarem o sistema de acordo com o usuário
- Diagrama de classes: utilizado desde o nível de análise até o nível de especificação. Mostra as classes e os dados que as mesmas possuem, seus métodos e como essas classes se relacionam e trocam informações;
- Diagrama de atividades: orientados ao fluxo de controle, descreve os passos para realizar uma determinada atividade. Sua notação pode representar ações paralelas, juntamente com sua sincronização.

Estes são apenas alguns dos diagramas que a UML possui e que estão a disposição para auxiliar no desenvolvimento de projetos, otimizando o processo de implementação, garantindo que não haja muita diferença entre a ideia original e o projeto final, devido ao surgimento de requisitos que podem aparecer ao longo do desenvolvimento.

# 3 SISTEMA WEB PARA CONTROLE DE ESTOQUE DE UMA CENTRAL DE COOPERATIVAS

Nesta parte do trabalho, são apresentados os resultados do levantamento e análise de requisitos, estes, que são de fundamental importância para o desenvolvimento do sistema.

## 3.1 Método

Trataremos agora das especificações do Sistema de controle de Estoque (CONTESCOOP), que foi proposto e desenvolvido neste trabalho. Será mostrado o levantamento de requisitos, diagramas de caso de uso e diagrama de classes, dando ênfase nas principais tarefas que o sistema deverá efetuar para alcançar os objetivos dos usuários e da organização como um todo.

O sistema de controle de estoque fora desenvolvido utilizando a linguagem de programação *Ruby*, visto que o mesmo além de ser um sistema de informação gerencial, segue também o conceito de sistema *web*, podendo assim ser utilizado de qualquer máquina, desde que tenha acesso a rede e seja autorizado previamente.

A metodologia de desenvolvimento utilizada foi o *Scrum*, devido a mesma, possuir características de acompanhamento do desenvolvimento diariamente, revisando e analisando o que foi projetado no dia anterior, determinando metas para os momento seguintes.

Pela sintaxe simples para usuários simples ou avançados, que busquem sua compreensão e não ser presa a um único paradigma, a escolha dessa linguagem foi adequado e para maximizar e agilizar o desenvolvimento do projeto. O *framework Ruby on Rails* foi escolhido, por levar em conta a quantidade de recursos, bibliotecas e materiais disponíveis, que juntos visam incentivar e fidelizar o grande número de desenvolvedores que tem aderido a arte de programar para *web*.

Nesse sistema, temos a possibilidade de cadastrar, cooperativas (no caso as filiadas a central, que podem ser devidamente definidas como singulares), cooperados (filiados as singulares, mas que, devido suas respectivas cooperativas serem filiadas à central, estes passam a serem sócios indiretos da central), produtos (informações sobre os mesmos, como também os lotes) que entram para formação do estoque que deve ser controlado, bem como as devidas saídas que ocorreram e o controle de usuário, visto que esse último é de suma importância para manuseio e segurança das informações inseridas e atualizadas no controle.

Além do f*ramework Ruby on Rails,* fora utilizado o *bootstrap*, que foi de fundamental importância para melhoramento da apresentação do sistema. Falamos de apresentação, nos referindo as telas, ícones e estrutura geral da página, conjunto de fatores estes, que juntos auxiliam e facilitam tantona utilização quanto no mesmo, devido a simbologia de cores e artefatos adequados.

## 3.2 Requisitos do Sistema

Como base para o desenvolvimento do sistema, tivemos a Central de Cooperativas dos Cajucultores do Estado do Piauí – COCAJUPI. Nela, foram realizadas reuniões com os funcionários responsáveis pela manutenção dos dados referentes ao controle de estoque, atividade esta, que até então eram feitas exclusivamente em planilhas eletrônicas. Esses funcionários citados são conhecidos no processo de análise de requisitos como *Stakeholders*, que são pessoas ou organizações que influenciam os requisitos do sistema.

Além das reuniões, pode-se aplicar questionário e analisar as devidas planilhas, onde ambos o resultado desses processos de análise, ocasionou em uma identificação das necessidades e dificuldades que os meios atuais oportunizavam. Foram questionados também, em relação às funcionalidades que o sistema deveria possuir.

No quadro 01, tem-se uma relação dos requisitos funcionais encontrados após as etapas citadas anteriormente. É também apresentado no quadro as dependências que cada requisito tem em relação a outro, caso seja necessária sua existência.

Quadro 01 – Requisitos Funcionais

| **Identificador** | **Descrição** | **Depende de** |
|---|---|---|
| RF01 | O sistema deverá possuir cadastro de usuários. | |
| RF02 | O usuário possuirá *login* e senha para acessar o sistema. | RF01 |
| RF03 | Os níveis de acesso aos recursos do sistema serão de acordo com os tipos. | RF02 |
| RF04 | O sistema permitirá cadastro de: cooperativas, cooperados e produtos. | RF03 |
| RF05 | O sistema permitirá entradas de produtos para formação de estoque. | RF04 |
| RF06 | O sistema permitirá saídas de produtos do estoque. | RF05 |
| RF07 | O sistema permitirá a geração de relatórios sobre as entradas no estoque. | RF05 |
| RF08 | O sistema permitirá a geração de relatórios sobre as saídas do estoque. | RF06 |

No quadro 02, são apresentados os requisitos não funcionais, com uma pequena descrição seguida por pelas categorias a qual pertencem e suas respectivas dependências.

Quadro 02 - Requisitos não funcionais

| **Identificador** | **Descrição** | **Categoria** | **Depende de** |
|---|---|---|---|
| RNF01 | O acesso as funcionalidades será realizado através das permissões que cada usuário possuirá, sendo estabelecido no cadastro. | Segurança de Acesso | RF02 |
| RNF02 | O sistema será utilizado na rede interna, podendo ser utilizado em redes externas, havendo devidas permissões para isso e compatibilidade com os navegadores a serem utilizados fora. | Portabilidade | |
| RNF03 | Emissão dos relatórios demonstrando resumidamentetodo o processo realizado pela organização em espaço de tempo estipulado. | Eficiência em relação ao tempo. | |
| RNF04 | Persistência das informações deve ser implementada em um Sistema Gerenciador de Bancos de Dados (SGBD) livre (*Postgres* ou MySQL). | Manutenabilidade. | |

O quadro 03 mostra as regras de negócios com seus respectivos identificadores, uma descrição e suas dependências.

Quadro 03 - Regras de negócio

| **Identificador** | **Descrição** | **Depende de** |
|---|---|---|
| RN01 | Só é permitida qualquer alteração no estoque, após cadastro da cooperativa, produtor e produto, respectivamente. | RF03; |
| RN02 | Todos os produtos deverão conter uma quantidade mínima, sendo inserida na hora do cadastro. | RF03; |
| RN03 | Deverá ser cadastrado uma data de vencimento para cada produto em cada uma de suas entradas que formarão estoque. | RF03; RF04 |
| RN04 | As saídas deverão obedecer e respeitar o estoque mínimo de cada produto. | RF03 |
| RN05 | Apenas um tipo de usuário poderá realizar cadastro de cooperativas e outros usuários. | RF02 |
| RN05 | O usuário de nível 1, apenas poderá controlar entradas e saídas da central, assim, seu acesso as outras cooperativas será apenas para leitura de dados. | RF03 |

Após o levantamento dos requisitos e regras de negócio no processo de análise e apresentados, esses puderam ser passados adiante na construção de diagramas, como falados na subseção 2.4.1, o de caso de uso, que devem demonstrar o que se pretende desenvolver de acordo com as atividades que os usuários terão disponíveis para suprir suas necessidades. Logo depois, serão descritos os diagramas de classe e posteriormente o de atividades.

## 3.3 Diagramas de Casos de Uso

O diagrama de caso de uso possibilita uma leitura e entendimento das funcionalidades que o sistema deverá possuir, para que quaisquer partes envolvidas no processo de desenvolvimento possa participar efetivamente do projeto, sem dúvidas e ou eventuais problemas por não terem uma base fundamental. Bezerra (2007) diz, “os diagramas de caso de uso, devem servir para dar suporte à parte escrita do modelo, fornecendo visão de alto nível, onde, quanto mais for a leitura, melhor.”

A figura 02 apresenta o caso de uso, onde são representadas as funcionalidades do sistema, bem como os atores envolvidos. Os atores que serão os usuários do sistema estão representados por bonecos, enquanto as funcionalidades, ou seja, os casos de uso estão representados por elipses e demonstrar as relações entre eles, as retas desempenham esse papel.

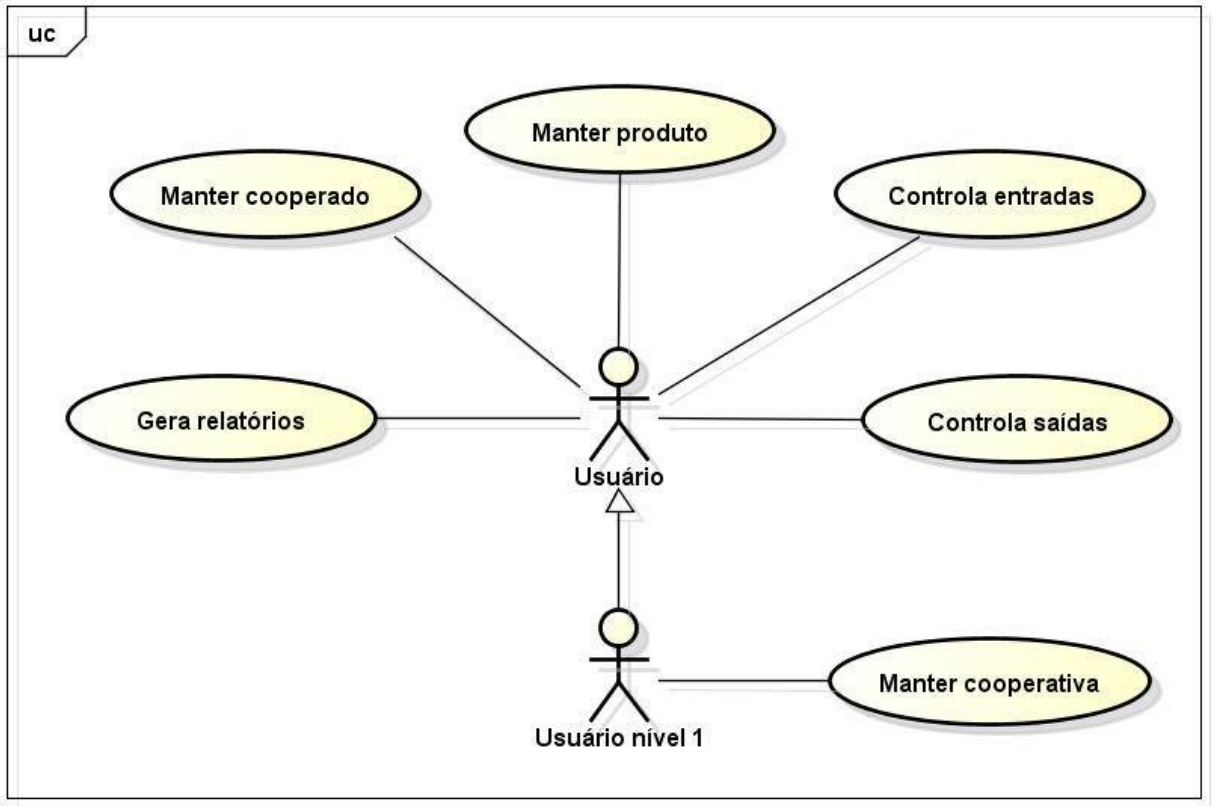



Figura 02 – Diagrama de Casos de uso

Fonte: O Autor (2015)

No diagrama mostrado na figura 02, podemos notar que o sistema possuirá dois usuários: o Usuário poderá desempenhar quase todas as funcionalidades do sistema, enquanto o Usuário nível 1, fará uso de todas e mais o cadastro de cooperativa. Isso se dá, pelo fato de restringir quem poderá utilizar as funcionalidades do sistema, ou seja, apenas pessoas autorizadas (cadastradas) por esse é que poderão realizar as demais funcionalidades restantes do sistema.

A seguir, especificações detalhando o diagrama de casos de usos, com seus atores envolvidos, requisitos e uma breve descrição, que auxiliará no entendimento de cada caso de uso.

- **Caso de Uso 01 – Manter usuário**
  - **Ator:** Usuário;
  - **Requisitos:** RF04;
  - **Descrição:** Realizar o cadastro, edição, exclusão e listagem dos usuários do sistema. Os usuários de nível 1, somente poderão ser cadastrados pelo Usuário, para garantir o controle e segurança no acesso as informações.
- **Caso de Uso 02 – Manter cooperativa**
  - **Ator:** Usuário nível 1;
  - **Requisitos:** RF04;
  - **Descrição:** Realizar o cadastro, edição, exclusão e listagem das cooperativas que compõem a central. As atividades ligadas a esse caso de uso, são exclusivamente realizadas pelo usuário nível 1, isso por que, as singulares compõem a central, logo apenas ela, pode realizar qualquer atividade em relação as cooperativas.
- **Caso de Uso 03 – Manter cooperado**
  - **Ator:** Usuário e Usuário nível 1;
  - **Requisitos:** RF04;
  - **Descrição:** Realizar cadastro, edição, exclusão e listagem dos cooperados das respectivas singulares. Essas ações podem ser realizadas por ambos os usuários devidos necessidade de acesso às informações dos mesmos.
- **Caso de Uso 04 – Manter produto**
  - **Ator:** Usuário e Usuário nível 1;
  - **Requisitos:** RF04;

- **Descrição:** Realizar cadastro, edição, exclusão e listagem dos produtos. Ambas as cooperativas podem realizar esse cadastro, pois nem sempre todas estão produzindo o mesmo produto.

- **Caso de Uso 05 – Controlar entradas**
  - **Ator:** Usuário e Usuário nível 1;
  - **Requisitos:** RF05;
  - **Descrição:** Controlar todas as entradas do sistema é sem dúvida umas das principais atividades a serem realizadas pelo sistema, para que haja um controle a rigor do estoque sendo formado, facilitando a obtenção das informações oriundas de determinado produto.
- **Caso de Uso 06 – Controlar saídas**
  - **Ator:** Usuário e Usuário nível 1;
  - **Requisitos:** RF06;
  - **Descrição:** Realizar o controle de todas as saídas de produtos do estoque, levando em conta o estoque mínimo e mantendo as informações referentes ao produto sempre atualizadas e seguras.
- **Caso de uso 07 – Gerar relatórios**
  - **Ator:** Usuário e Usuário nível 1;
  - **Requisitos:** RF08;
  - **Descrição:** A emissão programada ou sempre que necessária de relatórios, é uma atividade fundamental e de grande importância para acompanhamento de processos internos dentro de qualquer organização, assim, obter esses referentes a entradas, saídas, etc. é um fator crucial para o bom acompanhamento de dados da entidade.

Como o objetivo de uso do sistema é facilitar e melhorar os processos que envolvem o estoque de produtos da referida entidade, os casos de uso explicados, esclarecem que é uma atividade um pouco mais complicada que o normal, isso, porque cada entrada pode ser de vários cooperados diferentes, bem como também ocorre nas saídas, pois as informações levantadas podem ser de um grupo total.

Cada usuário estará ligado diretamente a uma cooperativa, diante disso, o cadastro das referidas unidades singulares e consecutivamente de seus usuários de nível 1, serão realizados pelo Usuário, que estará ligado diretamente a entidade central, sendo também o único que poderá realizar as ações de cadastrar o cooperado e a cooperativa.

## 3.4 Diagrama de Classes

A ser apresentado nesta seção, temos o Diagrama de Classes, que exibe as classes que compõem o sistema, juntamente com seus atributos, métodos e relacionamentos. Trata-se de um dos mais importantes diagramas da UML, onde se preocupa com a lógica do sistema, de como as classes se organizam, para facilitar o desenvolvimento de outros diagramas que poderão ser desenvolvidos.

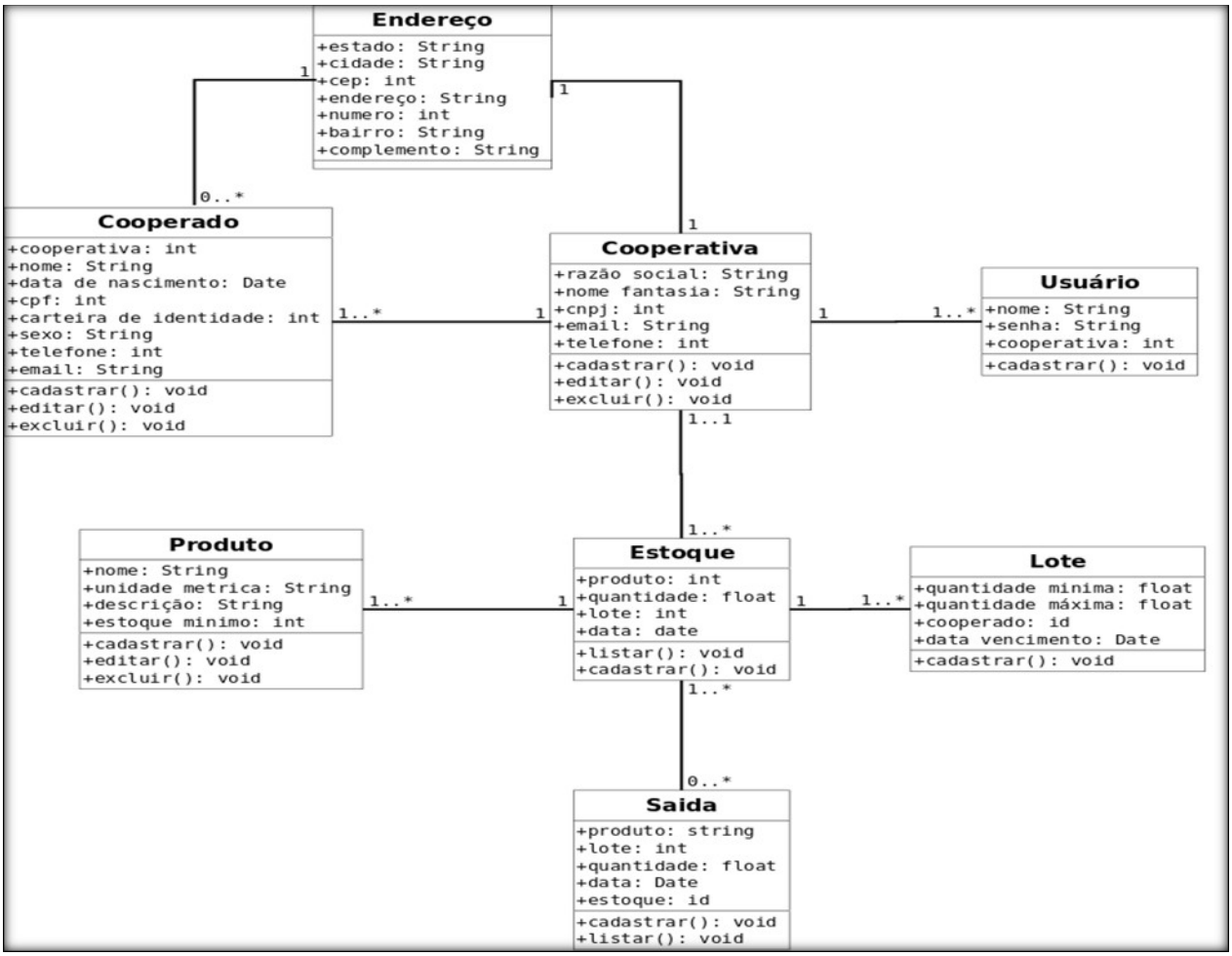
O sistema possuirá basicamente dois tipos de usuários, que no geral realizaram as mesmas atividades no sistema, com exceção da atividade de cadastrar cooperativa que será restrita ao usuário principal. Por isso, classe de usuário possui um relacionamento com a classe cooperativa.

A classe endereço possui relacionamentos com as classes cooperativa e cooperado, pois ambos devem informar endereços no cadastro, para manter a base de informações sobre eles. Foi considerado que, cada um deveria possuir no mínimo um e no máximo um endereço registrado, caso ocorra a necessidade de outro endereço, pode-se editar o endereço antigo, de forma a manter o mesmo sempre atualizado.

No caso das classes produto, lote e saídas, estas estão ligadas a classe estoque, pois toda alteração realizada nelas refletem no estoque gerado ou mantido pela cooperativa. Por exemplo, um produto é cadastrado apenas com informações sobre o mesmo, e isso, não representa uma entrada no estoque. Tendo o produto já cadastrado, há necessidade de criação do lote, para que feito isso, seja inserida uma entrada e assim, considerado a criação do estoque feita, a partir de então, a cooperativa possui o estoque de um determinado produto, ou mesmo vários.

A classe cooperativa poderá ter pelo menos um estoque, ou seja, uma quantidade relevante de um produto com lotes. Esses lotes são oriundos do cadastro do produto que foi repassado pelo cooperado, que necessariamente já faz parte da cooperativa e que só por meio do seu número de identificação (ID) dentro das tabelas do banco de dados, é que ocorre o rastreio do lote e consequentemente do produto.

A figura 03, abaixo, apresenta o diagrama de classe completo, com todas as tabelas que compõem o sistema, de forma a ilustrar o que foi esclarecido anteriormente.




Fonte: O Autor (2015)

Estes são apenas alguns dos diagramas que a UML possui e que estão a disposição para auxiliar no desenvolvimento de projetos, otimizando o processo de implementação, garantindo que não haja muita diferença entre a ideia original e o projeto final, devido ao surgimento de requisitos que podem aparecer ao longo do desenvolvimento.

# 4 RESULTADOS E DISCUSSÕES

Esse capítulo explica as principais funcionalidades do CONTESCOOP. Serão mostradas as funcionalidades que os tipos de usuários possuem acesso.

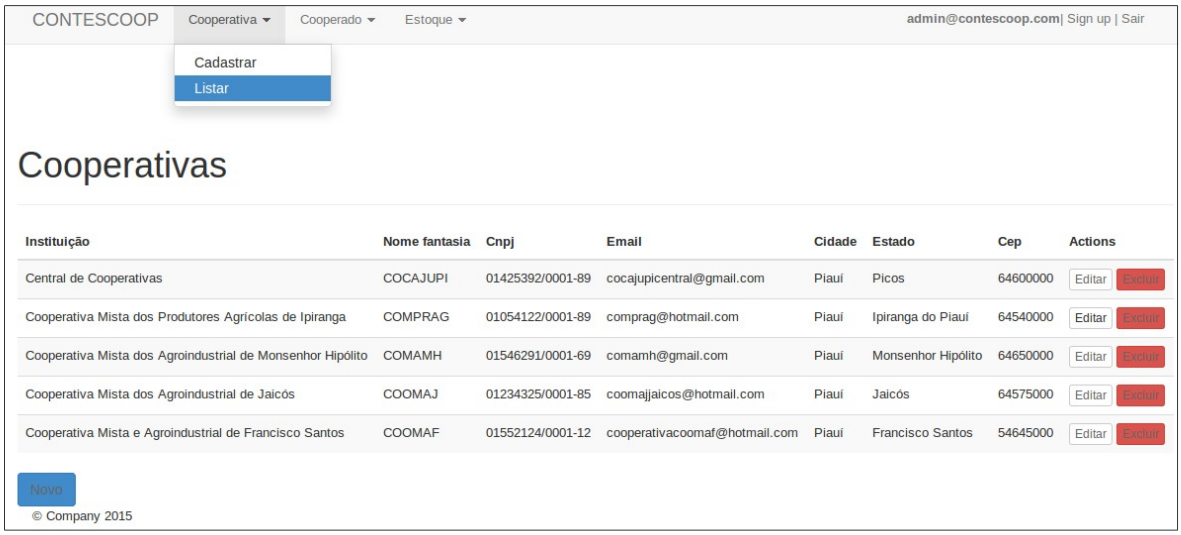
## 4.1 Funcionalidades

Para ter acesso as funcionalidades do sistema, é necessário que o usuário seja previamente cadastrado. Esse fato, garante que apenas as pessoas que possuem cadastro poderão acessar determinadas partes do sistema, bem como realizar apenas algumas tarefas. É o caso do usuário nível 1, ele terá acesso a todas as funções para cadastro de usuários, cooperativas e cooperados. Porém, não terá acesso ao cadastro de entrada de produtos, nas cooperativas singulares, ficando restritas apenas as entradas que acontecerão no estoque da central.

Devido o fato de cada singular e a central possuírem usuários distintos, o cadastro dos mesmos leva em conta a identificação da cooperativa em que o mesmo irá desempenhar durante o manejo do sistema.

Mesmo não podendo realizar o cadastrado dos devidos produtos no estoque das singulares, o usuário de nível 1, poderá ter acesso de leitura a esses dados, ou seja, poderá visualizar os produtos e suas respectivas quantidades no estoque formado em cada cooperativa, facilitando o acesso aos dados que até então exigiam certo trabalho para ser adquirido pela central.

Para acesso, o usuário possui um *login* e senha. De posse desses dados, todas as funcionalidades disponíveis para ela, aparecem no menu que estará sempre numa barra superior do sistema. A figura 04 mostra exatamente essas funcionalidades em seus respectivos menus.




**F**igura 04 – Funcionalidades do usuário nível 1.

Fonte: O Autor (2015)

Na figura 04, são apresentados as opções do menu cooperativa. Isso se deve ao fato de cada uma das cooperativas possuírem usuários diferentes. Nela o usuário logado é o de nível 1, que como falado anteriormente, pode cadastrar, editar e listar as cooperativas, além dos demais cadastros que o mesmo pode fazer, portanto, essa aba desse menu, só está disponível para este tipo de usuário.

Na figura 05, temos o usuário normal, que possui permissão para realizar todas as funcionalidades referentes a cooperativa, funções estas que envolvem diretamente o cadastro de produtores e processos referentes ao controle de estoque, onde está cadastrado, porém, não tem a funcionalidade de cadastrar outra cooperativa.


Figura 05 – Menu para usuários das singulares

Fonte: O Autor (2015)

No Caso de Uso 02, Manter Cooperativa, tem como uma das principais funcionalidades o cadastro das cooperativas que podem acessar o sistema, ou seja, aquelas que são filiadas a central. Além dessa funcionalidade que só pode ser efetuada pelo usuário de nível 1, há também as de editar os dados referentes a mesma, listagem de todas as singulares filiadas e a exclusão. Esta última, sempre que for executado, o usuário estará excluindo junto, todos os cooperados que compõem a mesma.

A figura 06 demonstra o cadastro de uma cooperativa pelo referido usuário. Como podemos notar, o cadastro da referida entidade, não necessita de informações previamente cadastradas no sistema, isso porque, cada uma representa uma ocorrência diferente, assim, cada uma possuirá seus cooperados, produtos, estoque, saídas e lotes distintos, sem levar em consideração o que outra singular possua.


Figura 06 – Funcionalidade Cadastrar Cooperativa.

Fonte: O Autor (2015)

A figura 07 mostra os cooperados cadastrados, com os botões de edição, exclusão e cadastro em determinada cooperativa. Essas funcionalidades fazem parte do Caso de Uso 03, Manter Cooperado. Nesse, todos os tipos de usuários podem realizar tais tarefas, desde que estejam devidamente logados na seção. Entende-se por seção o fato do mesmo estar utilizando a referência a cooperativa na qual o mesmo esta cadastrado, por um determinado espaço de tempo.

Para que seja possível o cadastro do cooperado, obrigatoriamente deve-se ter a cooperativa cadastrada, pois o mesmo deve conter no registro a entidade a qual é filiado, para devido controle de processo tanto por parte dos produtos que podem ser colocados a disposição da instituição, formando assim um estoque, quanto para saída do produto processado.

Deve-se ressaltar, que a exclusão de um cooperado, levará juntamente com ele, os lotes que foram cadastrados com a identificação do mesmo, isso porque, cada lote representa cada quantidade em datas e com produtos diferentes ou não, formam lotes diferentes para cada um dos sócios.

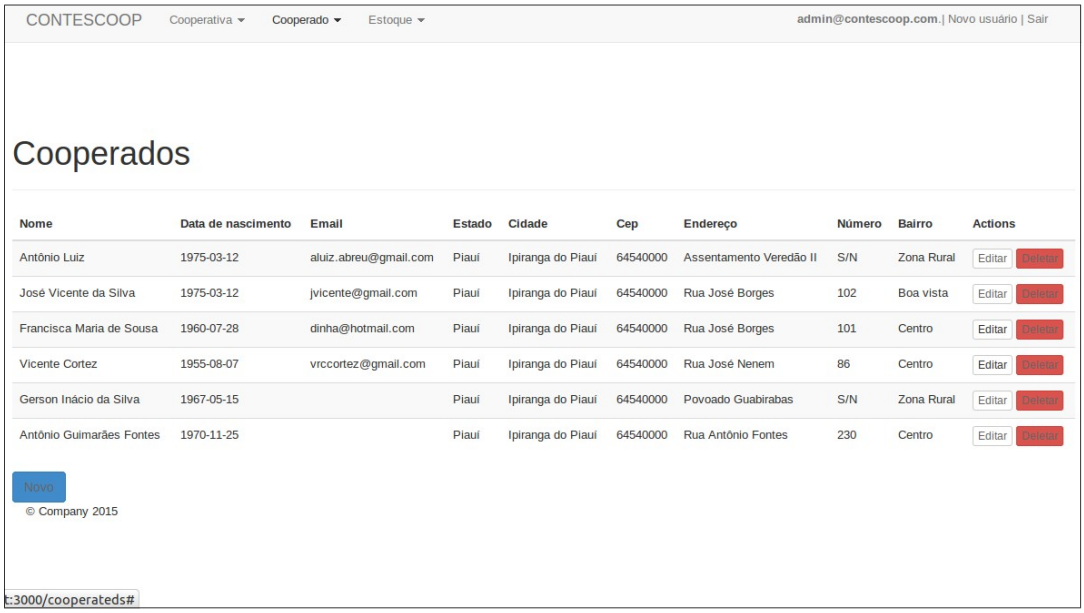



Figura 07 – Cooperados cadastrados. Ícones para edição, exclusão e cadastro.

Fonte: O Autor (2015)

Na figura 08, temos o usuário cadastrando um produto. Esse cadastro, necessariamente serve apenas para cadastro de informações referentes ao item. Essas informações serão utilizadas para cada entrada ou saída que ocorrerem no estoque, isto, porque, necessita-se do nome do produto, descrição, unidade métrica, para que aconteça a formação do estoque e no caso da saída, além dessas informações, é necessário o cuidado frequente quanto a quantidade mínima do produto que se encontra no estoque.

Uma quantidade mínima de determinado produto, garante que, sempre que o estoque estiver abaixo desse valor, o usuário deverá ser notificado, assim, o mesmo pode se prevenir para que nunca falte determinado produto.


Figura 08 – Funcionalidade de cadastro de produto.

Fonte: O Autor (2015)

Na descrição do Caso de Uso 05, são apresentadas as funcionalidades que mantém o estoque, deve-se levar em consideração as seguintes etapas: para Cadastrar uma entrada no estoque, o usuário deve inserir antes as informações referentes ao produto em questão; formar um lote, que possui quantidade mínima e máxima, bem como datas de entrada e vencimento do mesmo. Feita essas duas etapas respectivamente, o usuário poderá formalizar uma entrada no sistema, que obrigatoriamente, torna independentemente da quantidade, um estoque. Vale ressaltar que esse estoque, independe apenas da quantidade que é necessária para estoque, pois o usuário já tem previamente uma quantidade mínima e máxima cadastrada no lote e uma quantidade mínima do produto.

Na figura 09 mostra a criação de um lote, que como explicado anteriormente, deve ser realizado após o cadastro do produto e obrigatoriamente antes da inserção de uma quantidade de determinado elemento no estoque. O lote é obrigatoriamente cadastrado antes da entrada no estoque, devido a politica da instituição em questão, que determina que cada inserção de produtos no estoque, forma um novo lote, assim, um cooperado pode ter mais de um lote determinado produto, levando em consideração a data em que o produto foi depositado na instituição que formará o lote.


Figura 09 – Cadastro obrigatório do lote.

Fonte: O Autor (2015)

Na figura 10, demonstra-se o usuário inserindo produtos no estoque, caracterizando uma entrada. Assim, como os lotes já foram cadastrados, o usuário pode escolher um deles, de acordo com o produto que deseja inserir. Inseridas essas informações, pode-se informar a quantidade do produto e a data em que está sendo inserida. Nessa etapa, são inseridas apenas a quantidade, sem necessidade de escolha da unidade de medida, pois outrora já foi salva no ato de cadastro do produto.

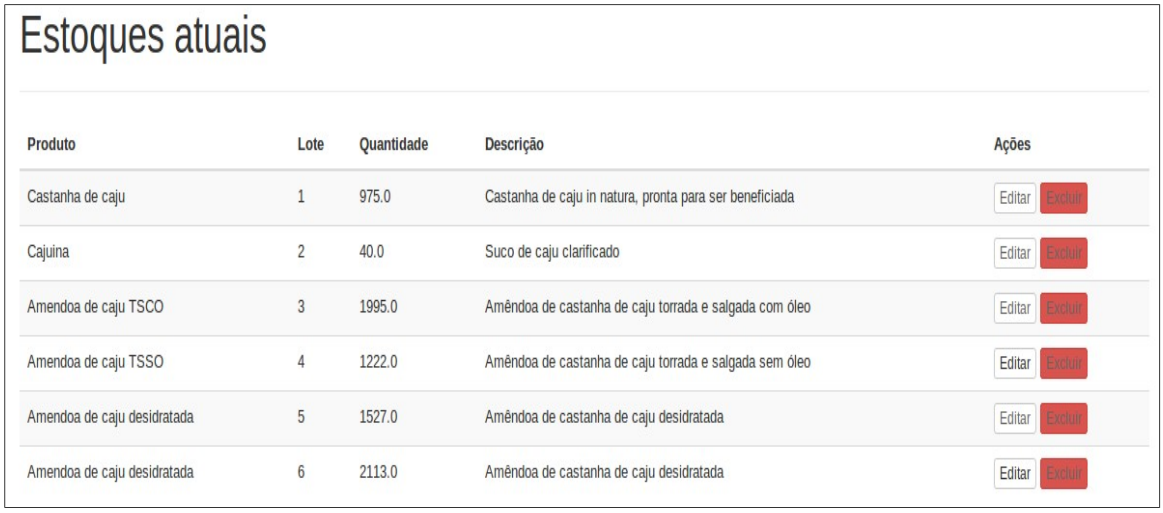

Figura 10 – Inserção de produtos no estoque, com lote.

Fonte: O Autor (2015)

Como parte do Caso de Uso, Manter Estoque, a listagem dos produtos cadastrados é parte essencial no sistema, isso, porque o usuário encontra as quantidades, descrições e número dos lotes de cada uma das formações de estoque, que ficam disponíveis para realização das operações de saída.

A necessidade de edição de uma ocorrência do estoque se dá, pelo fato de em raras exceções, um cooperado inserir várias quantidades de um mesmo produto, na mesma data e sem atingir o limite do lote. Assim, basta que o usuário efetue uma edição para atualizar esse valor.

A figura 11 representa a listagem de todos os produtos, lotes e demais informações que fazem parte das ocorrências de formação dos estoques, oportuniza ao usuário, acesso direto aos valores disponíveis para qualquer operação que venha a ser realizada posteriormente.




Fonte: O Autor(2015)

Todas as funções citadas, são parte fundamental para que o controle exato e seguro das informações referentes ao estoque que a central deve ter, estejam disponíveis. A utilização deste tipo de sistema, no caso web, proporcionou a disponibilidade das mesmas em qualquer máquina, visto que o usuário tendo seu cadastro previamente efetivado poderá ter acesso.

A saída de produtos do estoque é realizada com as informações do produto, que foram cadastradas previamente; o lote, que é utilizado como rastreio, isso porque, ao ser criado, são inseridas as informações referentes ao cooperado e uma data de vencimento para aquele lote.

Inseridas as informações são necessárias para dar baixa no estoque, pode ocorrer de não haver possibilidade de efetuar a operação. Isso, porque o estoque de determinado produto, pode ser inferior a quantidade pedida. A figura seguinte representa essa funcionalidade


Figura 12 – Funcionalidade que realiza baixa de produtos do estoque

Fonte: O Autor (2015)

Os usuários podem gerar relatórios. Essa funcionalidade está disponível para todos, devido a necessidade de cada uma das instituições precisarem de resumo das transações realizadas ao longo do mês, semestre ou ano. No caso do usuário nível 1, este pode gerar relatórios referentes as atividades das singulares, para que a central tenha sempre as informações de cada uma das cooperativas sempre atualizadas.

## 4.2 Resultados

Com a implantação do protótipo apresentado neste trabalho, foram resultados como a organização dos lotes com as devidas informações referentes a cada cooperativa e cooperados que as compõem.

As entradas e saídas de produtos dos estoques passaram a funcionar com mais rigor e precisão, devido o levantamento das informações que fazem parte desse processo, estarem disponíveis com praticidade, deixando esse trabalhado que até então era realizado manualmente, mais eficaz.

# 5 CONSIDERAÇÕES FINAIS

O presente trabalho apresentou o processo de desenvolvimento de um sistema de controle de estoque para uma central de cooperativa, assegurando que todas as informações necessárias para formação de estoque, estejam sempre disponíveis, segura e com facilidade de acesso.

Todos os passos seguidos para a obtenção do produto final foram apresentados. Esses levaram em consideração as necessidades que surgiram ao longo das análises e conhecimento do problema em questão, mas, que realizado todo o processo de obtenção de informações referentes ao mesmo, foram solucionados da melhor forma possível, visando atender e suprir as necessidades. Os métodos e tecnologias utilizadas foram apresentados e em seguida, as principais funcionalidades do sistema,

Como possíveis trabalhos futuros, têm-se objetivos de desenvolvimento para outras partes que agreguem outras informações, como administrativas, financeiras. Faz-se necessário ainda, tornar o sistema responsivo, para que possa ser utilizado em diversas telas, com tamanhos diferentes, adaptando-se, sem que seja necessário o usuário fazer esse processo manualmente.

## REFERÊNCIAS


BEZERRA, Eduardo. **Princípios de Análise e Projeto de Sistemas com UML**. 2 ed. Rio de Janeiro: Elsevier, 2007.

BURBECK, S. ***Applications Programming in model-view-controller*(mvc)**. 1992. Acesso em: 12 de Junho. 2015.

CAELUM, Apostila curso: **Desenv. Ágil para Web com RubyonRails, 2014**. Disponível em: < http://www.caelum.com.br/apostila-ruby-on-rails/ > Acesso em: 26 de julho de 2014.

CARVALHO, Adriano D. **O cooperativismo sob a ótica da gestão estratégica global**. 1.ed. [S.l]. Baraúna, 2011.

CRUZ, Tadeu. **Sistemas de Informações Gerenciais**. Atlas, São Paulo, Brasil, 2009.

FUENTES, Vinícius Baggio, ***RubyonRails*: coloque sua aplicação web nos trilhos**. 1.ed. São Paulo: Casa do Código, 2012.

FLANAGAN, David; MATSUMOTO, Yukihiro. **The Ruby Programming Language.** 1 ed. Sebastopol: O'Reilly Media, 2008.

GEHRKE, Johannes; RAMARKRISHNAN, Raghu; TANIWAKE, Célia, **Sistemas de gerenciamento de banco de dados**. 3.ed. [S.l]. Gisélia Costa, 2011.

GORDON, Steven R; GORDON, Judith R. **Sistemas de informação**: uma abordagem gerencial. Rio de Janeiro: LTC, 2006.

GUEDES, Gilleanes T. A. **UML 2 Uma abordagem prática**. 2 ed. São Paulo: Novatec, 2011.

LAUDON, Kenneth C; LAUDON, Jane P. **Sistemas de Informação Gerenciais** Administrando a empresa digital. 5 ed. São Paulo: Pearson Prentice Hall, 2004.

LOBO, Edson Junio Rodrigues, **Curso prático de MSQL.** 1.ed. [S.l].Digerati Books, 2008.

MAGNO, Alexandre. **Mobile First Bootstrap** Develop advanced websites optimized for mobile devices using the Mobile First feature of Bootstrap**.** 1 ed. Birminghan: Packt Publishing, 2013.

MELO, Ana Cristina Vieira de; SILVA, Flávio Soares Corrêa da. **Princípios de Linguagens de Programação**. Edgard Blucher, São Paulo, Brasil, 2003.

MUTO, Claudio Adonai, **PHP & MYSQL – Guia introdutório.** 3.ed. [S.l].Brasport, 2006.

NUNES, Mauro; O'NIELL, Henrique. **Fundamental de UML**. 3 ed. FCA – Editora de Informática, 2013.

O'BRIEN, James A. **Sistemas de informação e as decisões gerencias na era da internet.** 2 ed. São Paulo: Saraiva, 2006.

OLIVEIRA, Eustáquio R, **Ruby - Conhecendo a Linguagem**. 1.ed.Rio de Janeiro :Brasport, 2006.

PAULI, Josh. **Introdução ao web hacking Ferramentas e técnicas para invasão de aplicações web.** 1 ed. São Paulo: Novatec, 2013.
PRESSMAN, Roger S. **Engenharia de Software: uma abordagem profissional**. 7 ed. Porto Alegre: AMGH, 2011.

REZENDE, Denis A. **Sistemas de informações organizacionais: guia prático para projetos em cursos de administração, contabilidade e informática**. 2 ed. Atlas, 2007.

RODRIGUES, Andréa. **Desenvolvimento para Internet** Curitiba: LT, 2010.

SABBAGH, Rafael. **SCRUM:** Gestão ágil para projetos de sucesso. São Paulo: Casa do Código, 2013.

SILVA, Maurício S. **Criando sites com HTML**: sites de alta qualidade com HTML e CSS. Novatec, São Paulo, Brasil, 2008.

SOUZA, Lucas. **Ruby: Aprenda a programar na linguagem mais divertida**. São Paulo: Casa do Código, 2012.

TONSIG, Sérgio L. **Análise e Projeto de Sistemas I**. 1 ed. São Paulo: Faculdade de Tecnologia de Alta Noroeste, 2000.

VELLOSO, Fernando de Castro. **Informática conceitos básicos**. Elsevier, Rio de Janeiro, Brasil, 2003.

XEXÉO, Geraldo. **Modelagem de Sistemas de Informação: Da análise de requisitos ao modelo de interface.** *Creative Commons*, 2007.

**TERMO DE AUTORIZAÇÃO PARA PUBLICAÇÃO DIGITAL NA BIBLIOTECA "JOSÉ ALBANO DE MACEDO"**

**Identificação do Tipo de Documento**

( ) Tese

( ) Dissertação

(X) Monografia

( ) Artigo

Eu, Felipe Iuran de Sousa,

autorizo com base na Lei Federal nº 9.610 de 19 de Fevereiro de 1998 e na Lei nº 10.973 de 02 de dezembro de 2004, a biblioteca da Universidade Federal do Piauí a divulgar, gratuitamente, sem ressarcimento de direitos autorais, o texto integral da publicação Sistema Web para controle de estoque de uma central de cooperativas - ConTescoop.

de minha autoria, em formato PDF, para fins de leitura e/ou impressão, pela internet a título de divulgação da produção científica gerada pela Universidade.

Picos-PI 05 de dezembro de 2016.

Felipe Iuran de Sousa

Assinatura

Felipe Iuran de Sousa

Assinatura