UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO DE ESPECIALIZAÇÃO EM TECNOLOGIA JAVA

RAFAEL OLIVEIRA

SISTEMA WEB PARA GERENCIAMENTO DE MANUTENÇÕES EM MÁQUINAS INDUSTRIAIS

MONOGRAFIA DE ESPECIALIZAÇÃO

PATO BRANCO
2015

RAFAEL OLIVEIRA

SISTEMA WEB PARA GERENCIAMENTO DE MANUTENÇÃO EM MÁQUINAS INDUSTRIAIS

Trabalho de Conclusão de Curso, apresentado ao III Curso de Especialização em Tecnologia Java, do Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, como requisito parcial para obtenção do título de Especialista.

Orientadora: Profa. Beatriz Terezinha Borsoi

PATO BRANCO
2015

SISTEMA WEB PARA GERENCIAMENTO DE MANUTENÇÕES EM MÁQUINAS INDUSTRIAIS

Por

Rafael Oliveira

Esta monografia foi apresentada às 14h00 do dia 15 de outubro de 2015 como requisito parcial para a obtenção do título de ESPECIALISTA, no III curso de Especialização em Tecnologia Java, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. O acadêmico foi arguido pela Banca Examinadora composto pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Profa. Dra. Beatriz Brsoi
Orientadora
UTFPR – Campus Pato Branco

Profa. Msc. Lucilia Yoshie Araki
Banca
UTFPR – Campus Pato Branco

Prof. Msc. Robison Cris Brito
Banca
UTFPR – Campus Pato Branco

Prof. Msc. Robison Cris Brito
Coordenador do curso de Especialização
UTFPR – Campus Pato Branco

RESUMO

OLIVEIRA, Rafael. Sistema web para gerenciamento de manutenções em máquinas industriais. 33 f. Monografia (Trabalho de especialização) – Departamento Acadêmico de Informática, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2015.

A manutenção de máquinas industriais tem como principal objetivo manter os equipamentos em condições de pleno funcionamento. Se ocorrer parada da máquina em decorrência de quebra, por exemplo, é necessário que a mesma seja colocada em funcionamento o mais rapidamente possível, visando que o impacto na produção seja minimizado. Visando reduzir as paradas não previstas e o impacto dessas paradas existem as manutenções preventivas. Contudo, independentemente do motivo do reparo ou conserto é importante que o mesmo seja realizado o mais rapidamente possível e com o menor custo. Um aplicativo computacional para o gerenciamento de manutenções em máquinas industriais visa promover agilidade e otimização de recursos, representados por peças e outros e o trabalho das pessoas que é o serviço envolvido na manutenção. Um aplicativo com esse objetivo foi desenvolvido como resultado deste trabalho. É uma aplicação para *web* desenvolvida utilizando a linguagem Java e tecnologias associadas ao desenvolvimento de aplicações *web* e com o banco de dados PostgreSQL.

Palavras-chave: Aplicativo para gerenciamento de manutenções de máquinas industriais. Aplicativo web. Aplicação interface rica.

LISTA DE FIGURAS

LISTAGENS DE CÓDIGOS

LISTA DE SIGLAS

CRUD	*Create, Read, Update and Delete*
CSS	*Cascading Style Sheets*
HTML	*HyperText Markup Language*
JPA	*Java Persistence API*
JSON	*JavaScript Object Notation*
OS	Ordem de Serviço
RF	Requisitos Funcionais
RNF	Requisitos Não Funcionais
URL	*Uniform Resource Identifier*

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta as considerações iniciais, os objetivos e a justificativa. As considerações iniciais apresentam o contexto no qual se insere o aplicativo desenvolvido como resultado deste trabalho. O capítulo é finalizado com a apresentação dos capítulos subsequentes.

1.1 CONSIDERAÇÕES INICIAIS

A manutenção de máquinas industriais pode ser motivada por diversos fatores resumidos em duas finalidades básicas, sendo a correção e a prevenção. A necessidade de manutenção em máquinas não ocorre apenas no segmento industrial, mas este é citado por ser o escopo deste trabalho. Embora as finalidades básicas das manutenções realizadas em máquinas sejam de manutenção e prevenção, há outras classificações como as propostos por Slack *et al.* (2002) e Pinto e Xavier (2003). Para Slack *et al.* (2002) há as categorias: corretiva, preventiva e preditiva. E para Pinto e Xavier (2003), as manutenções são: corretiva não planejada, corretiva planejada, preventiva, preditiva, detectiva e engenharia de manutenção.

A manutenção corretiva visa colocar novamente em condições de uso máquinas que por algum motivo estão parcial ou totalmente inativas para a produção. A prevenção tem como objetivo evitar futuras paradas em decorrência de problemas que possam ocasionar a necessidade de consertos.

As manutenções de qualquer natureza realizadas em uma indústria são, geralmente, gerenciadas por meio de ordens de serviço. Essas ordens identificam a máquina que é objeto de conserto e o serviço a ser realizado. A partir dessa identificação é possível definir os profissionais envolvidos no serviço, o tempo, as peças e os equipamentos necessários para realizar a manutenção. Além disso, calcular os custos para a realização do serviço e o valor a ser gasto com os materiais necessário. E, também, estimar o prejuízo ocasionado pela parada da máquina e a necessidade de medidas para manter, de alguma forma em atividade, o serviço realizado pela máquina parada.

Um aplicativo computacional para o gerenciamento dessas ordens permite acompanhar os serviços que precisam ser realizados, as máquinas em manutenção e o agendamento das manutenções preventivas. E, ainda, o gerenciamento dos gastos com

manutenção de cada máquina é mais facilmente identificado por meio de um sistema que permita e facilite esses registros.

É nesse contexto que se insere o aplicativo desenvolvido com a realização deste trabalho. Com o uso do aplicativo, que é para *web*, as ordens de serviço são mais facilmente gerenciadas e o acompanhamento do estado das mesmas também é facilitado.

1.2 OBJETIVOS

O objetivo geral e os objetivos específicos deste trabalho são apresentados a seguir.

1.2.1 Objetivo Geral

Gerenciar, por meio de um sistema *web*, as ordens de manutenção de máquinas industriais.

1.2.2 Objetivos Específicos

- Desenvolver um sistema para realizar o controle de manutenções preventivas de máquinas industriais.
- Gerenciar as manutenções periódicas por meio de informes da proximidade da data de sua realização.
- Gerenciar os gastos de manutenção de cada máquina industrial visando identificar a viabilidade de manutenção da máquina ou a necessidade de substituição da mesma.

1.3 JUSTIFICATIVA

A infraestrutura das indústrias é composta por um conjunto de máquinas. E se a indústria é de grande porte o número dessas máquinas pode ser consideravelmente grande. Para o funcionamento adequado dessas máquinas e também pela segurança da produção são realizadas manutenções, além dos consertos, que visam prevenir quebras que ocasionam

prejuízos. Quanto maior o número de máquinas maior a dificuldade de controle desses itens, das datas de manutenções realizadas e planejadas, dos custos e do pessoal envolvido no processo.

O uso de um aplicativo computacional *web* facilita o registro e o acompanhamento das ordens de serviço pendentes, em realização e as já realizadas. O registro pode ser realizado no momento da ocorrência e os dados para acompanhamento podem ser obtidos a partir de qualquer computador que estiver na rede.

O aplicativo desenvolvido possibilitará maior controle das manutenções realizadas e o valor total gasto em manutenção de cada máquina permitindo, assim, avaliar a viabilidade de manutenção da máquina em relação ao seu impacto na linha de produção e de custos e benefícios.

1.4 ESTRUTURA DO TRABALHO

O Capítulo 2 apresenta o referencial teórico do trabalho e está fundamentado em manutenção de equipamentos industriais. No Capítulo 3 são apresentados os materiais e o método utilizados para o desenvolvimento do trabalho. Os resultados são apresentados no Capítulo 4. Por fim estão as considerações finais seguidas das referências bibliográficas utilizadas na composição do texto.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta a fundamentação conceitual do sistema desenvolvido como resultado deste trabalho. O sistema visa o gerenciamento de manutenções em máquinas industriais. Assim, o referencial teórico trata sobre a manutenção dessas máquinas e equipamentos.

2.1 MANUTENÇÃO EM EQUIPAMENTOS INDUSTRIAIS

A manutenção pode ser entendida como os cuidados técnicos realizados e que são indispensáveis ao adequado funcionamento de máquinas, equipamentos, ferramentas e instalações (PINTO; XAVIER, 2001). Para esses autores, esses cuidados envolvem a conservação, a adequação, a restauração, a substituição e a prevenção.

A importância da manutenção é ressaltada na evidência que a disponibilidade de funcionamento dos equipamentos, principalmente dos mais relevantes em um processo de produção, é um fator importante para a qualidade de produtos, o cumprimento de prazos e a lucratividade da empresa. O desempenho inadequado das máquinas, a manutenção ineficaz e tempos de manutenção elevados conduzem ao aumento de custos de produção, perdas de mercado e de oportunidades, além de redução de lucros e a existência de outros resultados não desejáveis para a empresa (CAPETTI, 2005).

Slack *et al.* (2002) classificam as manutenções em três tipos básicos: corretiva, preventiva e preditiva. Pinto e Xavier (2003), por outro lado, apresentam seis como os tipos básicos de manutenções: corretiva não planejada, corretiva planejada, preventiva, preditiva, detectiva e engenharia de manutenção. A seguir são apresentados os conceitos relacionados a esses tipos de prevenções de acordo com as definições constantes em Slack *et al.* (2002), Pinto e Xavier (2003) e Capetti (2005):

a) Manutenção corretiva - realizada após a ocorrência de falha. A manutenção corretiva pode ser planejada quando visa intervir na ocorrência de desempenho inferior ao esperado e pode haver a decisão de manter o funcionamento da máquina ou equipamento até a sua parada definitiva (SILVA, 2004).

b) Manutenção preventiva - é a atuação realizada visando minimizar ou evitar ocorrências que possam causar parada na produção ou redução de desempenho. Essas manutenções são realizadas seguindo um plano previamente definido e elas podem ser

periódicas em termos de intervalo de tempo, tempo de uso da máquina ou equipamento, horas de uso ou outro critério. Esse tipo de manutenção visa impedir a ocorrência de falhas e manter o controle dos equipamentos realizando operações presumidas e consideradas convenientes para o trabalho realizado por cada máquina (SILVA; ANTUNES, 2012). Nesse tipo de manutenção são realizados ajustes e/ou substituições de peças, sempre de maneira preventiva.

c) Manutenção preditiva - visa monitorar o desgaste de componentes das máquinas, para que sejam substituídos quando for necessário. Esse tipo de manutenção visa evitar o risco de paradas repentinas em máquinas. Contudo, a realização de manutenção preditiva não justifica a interrupção do trabalho de máquinas para ajustes ou trocas de peças desnecessariamente.

d) Manutenção detectiva - é a atuação que é efetuada com o objetivo de detectar falhas que não foram identificadas ou ainda não são perceptíveis pela equipe de operação e de manutenção.

e) Engenharia de manutenção - visa modificar situações permanentes de desempenho. De acordo com Xavier (2015, p. 3), engenharia de manutenção “é o conjunto de atividades que permite que a confiabilidade seja aumentada e a disponibilidade garantida” das máquinas e equipamentos.

Silva (2004) destaca que para integrar todos os processos envolvidos na manutenção é fundamental a existência de um sistema de controle que permita a realização de determinados atividades e a identificação de determinadas informações:

a) que serviços serão realizados;

b) quando os serviços serão realizados;

c) quanto tempo será utilizado na realização de cada serviço;

d) qual é o custo de cada serviço, o custo acumulado de cada máquina e o custo total das máquinas da indústria, da linha de produção ou outro agrupamento;

e) quais os materiais (componentes, peças e outros) necessários para realizar a manutenção;

f) quais os profissionais necessários para realizar o serviço;

g) quais as máquinas, equipamentos e ferramentas serão necessários para realizar a manutenção.

3 MATERIAIS E MÉTODO

Este Capítulo apresenta os materiais e o método utilizados para a modelagem e o desenvolvimento do aplicativo obtido como resultado da realização deste trabalho.

3.1 MATERIAIS

O Quadro 1 apresenta as tecnologias e as ferramentas utilizadas para a modelagem e a implementação do sistema.

Tecnologia	Versão	Referência	Finalidade
Java Plataform (JDK)	1.8	http://www.oracle.com/	Linguagem de programação *backend*
PostgreSQL	9.3	http://www.postgresql.org/	Banco de Dados
pgAdmin	1.16.1	http://www.pgadmin.org/	Administrado do banco de dados
BootStrap	3.3.4	http://getbootstrap.com/	Para formatação do *front-end*
Spring Data	1.8.0	http://projects.spring.io/spring-data-jpa/	*Framework* objeto-relacional
Spring Security	4.0.0	http://projects.spring.io/spring-security/	Segurança da aplicação
Spring Boot	1.2.6	http://projects.spring.io/spring-boot/	Configuração do projeto Spring
Eclipse	4.4.0	http://www.eclipse.org/	Ambiente de desenvolvimento
AngularJs	1.2.29	https://angularjs.org/	Extensão para *HyperText Markup Language* (HTML)
Maven	3.3.3	https://maven.apache.org/	Gerenciamento das dependências do projeto

Quadro 1 – Ferramentas e materiais

3.2 MÉTODO

Como método de desenvolvimento, a ideia básica do processo iterativo e incremental (KRUCHTEN, 2004) foi utilizada. Não foram realizados todos os processos indicados nesse método, mas a ideia de desenvolver entregáveis que pudessem ser utilizados à medida que as

funcionalidades eram implementadas. Como modelo de ciclo de vida para a definição das atividades ou processos foi utilizado o modelo sequencial linear de Pressman (2006).

A seguir são apresentadas as principais atividades desenvolvidas, agrupadas de acordo com as fases do modelo sequencial linear de Pressman (2006). Contudo, ressalta-se que essas fases tiveram iterações, como preconizado pelo modelo iterativo e incremental (KRUCHTEN, 2004).

a) Requisitos

A definição das funcionalidades para o sistema foi realizada a partir de demanda de uma indústria na área de metalurgia. A obtenção dos requisitos foi realizada a partir das necessidades indicadas pelo gerente de manutenção dessa indústria. A coleta dos requisitos foi realizada a partir de conversas informais com esse gerente de manutenção e pela observação do processo realizada. Os seguintes passos foram realizados:

1) Identificação dos envolvidos e as tarefas realizadas no processo de manutenção em máquinas da indústria. Da identificação dos envolvidos foram identificados os atores como usuários do sistema: administrador (admin) e usuário (user). O primeiro com acesso a todas as funcionalidades do sistema e o segundo com acesso a inclusão de dados cadastrais e de ordens de serviço.

2) Identificação de problemas enfrentados atualmente no registro e acompanhamento das manutenções realizadas. Problemas foram identificados no procedimento atual de registro das ordens de serviço. A partir da análise realizada uma proposta de procedimento foi realizada e que passou a ser adotada pela equipe envolvida no registro das ordens de serviço, encaminhamento e acompanhamento do serviço de manutenção.

3) Identificação de um novo fluxo no processo de manutenção visando melhorias no atual em decorrência do sistema desenvolvido. Os problemas identificados auxiliaram a redefinir o fluxo de trabalho adotado na empresa. A principal alteração no fluxo está relacionada a quem lança os itens na ordem de serviço. Antes o próprio técnico quem preenchia a ficha, com a análise realizada para o levantamento dos requisitos do trabalho, passou a ser o gerente de manutenção acompanha e lança na ordem de serviço.`

b) Modelagem

Após realizado o levantamento de requisitos eles foram organizados em funcionais e não funcionais, servindo de base para a definição dos dados a serem armazenados e a elaboração dos formulários de cadastro e consulta.

c) Implementação

Inicialmente o desenvolvimento foi realizado utilizando JSP, mas ao decorrer da implementação surgiram dificuldades pelo fato de a tecnologia não facilitar a implementação de funcionalidades que permitissem dinamismo ao sistema dinâmico. Por exemplo o lançamento de itens na ordem de serviço na qual é adicionada uma linha para cada item acrescentado. Optando pelo *framework* Angular no *front-end* esse problema foi resolvido. Outros benefícios do *framework* no desenvolvimento do projeto estão relacionados à possibilidade de manter a aplicação como cache de arquivos HTML, o que facilitou a padronização de desenvolvimento.

d) Testes

Os testes foram informais e realizados pelo desenvolvedor. Quando implementado um grupo de requisitos que definisse uma funcionalidade de negócio do sistema, após realizados os testes unitários para identificar erros de codificação, a versão era encaminhada para futuros usuários, de maneira que pudessem verificar se as funcionalidades do negócio estavam sendo atendidas.

4 RESULTADOS

Este capítulo apresenta o aplicativo desenvolvido como resultado deste trabalho. Na Seção 4.1 é apresentada uma visão geral das funcionalidades e do escopo do sistema. Essas funcionalidades são apresentadas e modeladas como requisitos na Seção 4.2. A apresentação das funcionalidades do aplicativo é realizada por meio das suas telas com explicação dos recursos na Seção 4.3. A Seção 4.4 estão partes do código implementado com objetivo de mostrar o uso da tecnologia no desenvolvimento dos requisitos do aplicativo.

4.1 ESCOPO DO SISTEMA

O sistema tem o papel de auxiliar no controle de manutenções das máquinas de uma indústria, quando ocorre algum problema de funcionamento na máquina o gerente de produção abre uma ordem de serviço com uma breve descrição do problema que é listada na tela principal do sistema para o gerente de manutenção, ele avisa o técnico que vai levantar o problema, esse tempo que o técnico leva será lançado como serviço na Ordem de Serviço (OS). No final do mês é possível tirar relatórios para saber a quantidade de horas trabalhadas em cada máquina e também as peças utilizadas gerando um custo na produção.

As máquinas serão primeiramente cadastradas e o gerente de manutenção abrirá ordens de serviço quando houver alguma máquina com problema. Nessa ordem de serviço serão contabilizadas as horas de trabalho do técnico e as peças necessárias para o funcionamento da mesma.

O operador também faz os agendamentos para a manutenção preventiva das máquinas. O sistema fará o lembrete dessas manutenções por meio de avisos. Os produtos e serviço são previamente cadastrados e vinculados a grupos para se totalizar os custos, por exemplo, por: parte elétrica, parafusos e rolamentos, assim, é possível ter um controle mais detalhado dos custos.

4.2 MODELAGEM DO SISTEMA

O Quadro 2 apresenta os Requisitos Funcionais (RF) definidos para o sistema. Os requisitos são definidos como ações realizadas por meio das funcionalidades providas pelo sistema.

Identificação	Ação	Descrição
RF01	Cadastrar grupos	Cada item cadastrado está associado a um grupo. A organização por grupos visa facilitar a filtragem em relatórios, por exemplo. Exemplo de grupos: serviços elétricos, parafusos.
RF02	Cadastrar itens	Cadastrar os itens que podem ser produtos ou serviços. Ambos são utilizados na composição das ordens de serviço.
RF03	Cadastrar máquinas	Cadastrar máquinas presentes na industria para possibilitar o lançamento de OS onde elas são vinculadas.
RF04	Cadastrar funcionários	Funcionários são gerentes de manutenção e técnicos que prestam serviços de manutenção nas máquinas. Os funcionários são designados para realizar as ordens de serviços.
RF05	Abrir ordem de serviço	Quando identificado algum problema ou for necessário realizar uma manutenção na máquina.
RF06	Movimentar ordem de serviço	O técnico responsável pela realização da ordem de serviço lançará os itens necessários.
RF07	Fechar ordem de serviço	O fechamento é realizado quando o serviço na máquina é finalizado.
RF08	Consultar serviços realizados	Consulta de todos os serviços realizados dentro do período filtrados por máquina.

Quadro 2 – Requisitos funcionais

Os Requisitos Não Funcionais (RNF) definidos para o sistema são apresentados no Quadro 3.

Identificação	Ação	Descrição
RNF01	Acesso ao sistema	O acesso ao sistema será realizado por meio de *login* e senha previamente cadastrados.
RNF02	Lançamento de ordens de serviço	Somente os manutentores poderão lançar novas ordens de serviços.
RNF03	Movimentação da ordem de serviço	O técnico que realizar os serviços repassará para o manutentor movimentar a ordem de serviço.

Quadro 3 – Requisitos não funcionais

A Figura 1 apresenta o diagrama de entidades e relacionamentos do banco de dados. As entidades principais são ordens de serviço que são compostas por itens que pertencem a grupos. As ordens de serviço estão relacionadas às máquinas. As demais tabelas representadas no diagrama estão relacionadas aos usuários do sistema e às respectivas permissões de acesso.

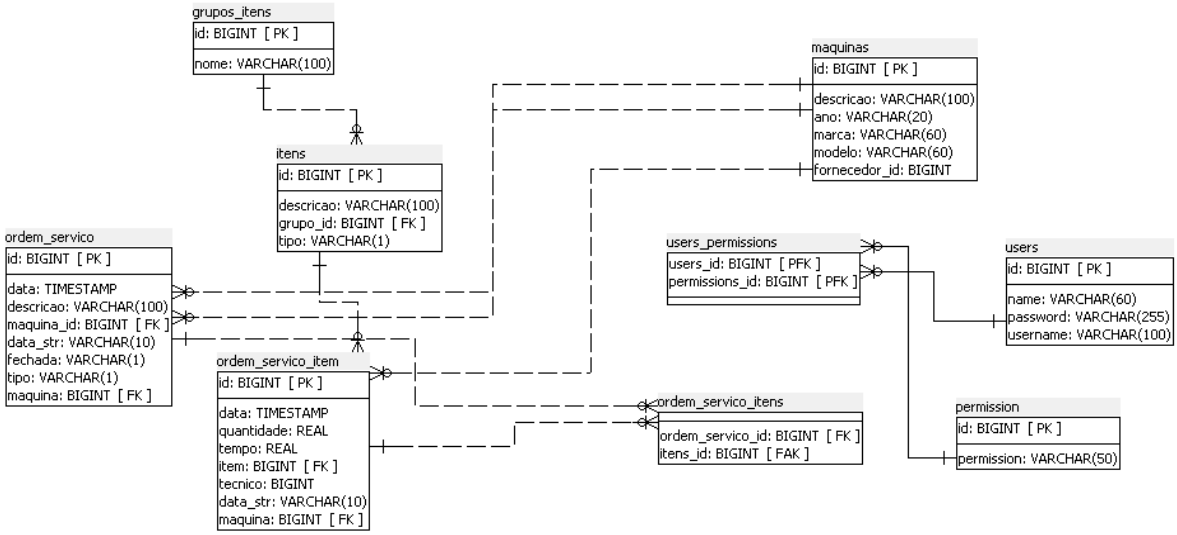

Figura 1 – Diagrama de entidades e relacionamentos do banco de dados

4.3 APRESENTAÇÃO DO SISTEMA

É necessário informar *login* e senha para ter acesso ao sistema. O sistema reconhece o perfil do usuário e atribui as permissões definidas a ele que são as seguintes:

a) administrador (admin) – esse tipo de usuário tem acesso a todas as funcionalidades do sistema;

b) usuário (user) – esse tipo de usuário tem acesso a todos os cadastros podendo incluir e alterar o único cadastro que ele não pode alterar é o da OS.

O perfil que geralmente atualiza as informações de cadastros no sistema é o *user*, por ser um usuário mais comum. Para realizar o lançamento de itens na OS e consultar os relatórios somente o perfil de administrador poderá fazê-los.

O sistema tem seu ciclo de vida da seguinte maneira:

a) são cadastrados os grupos dos itens de acordo com o detalhamento que a industria necessita conforme a Figura 2.

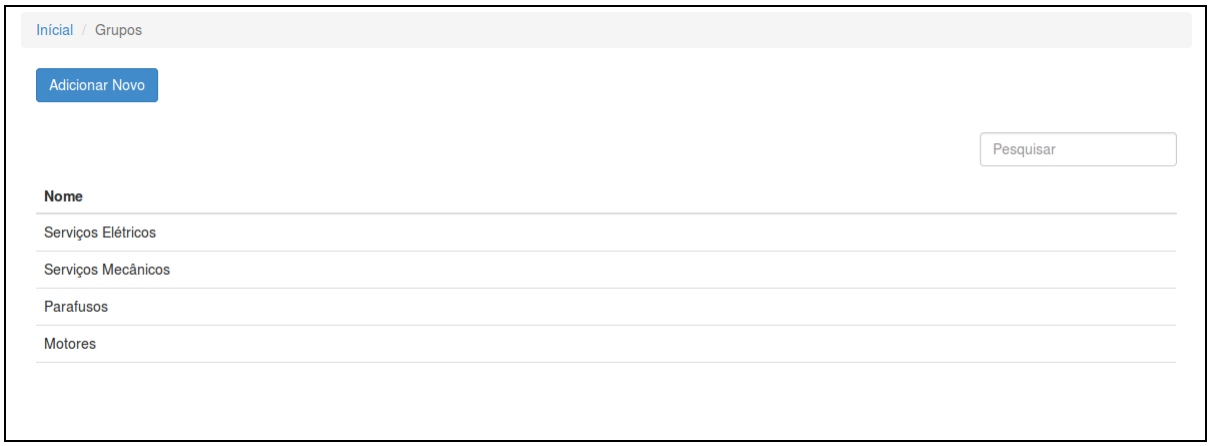

Figura 2 – Cadastro de grupos de itens

Após lançados os itens e os serviços que estão disponíveis, são cadastrados todas as peças utilizadas e também os serviços realizados nas máquinas.

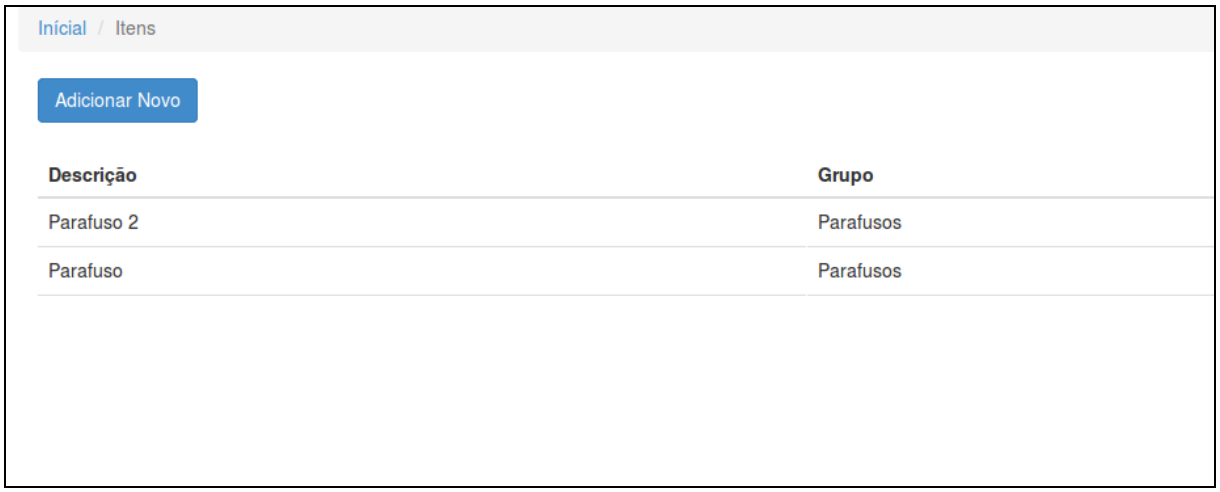

Figura 3 – Listagem de itens

Na Figura 4 está a tela para lançamento de um novo item.

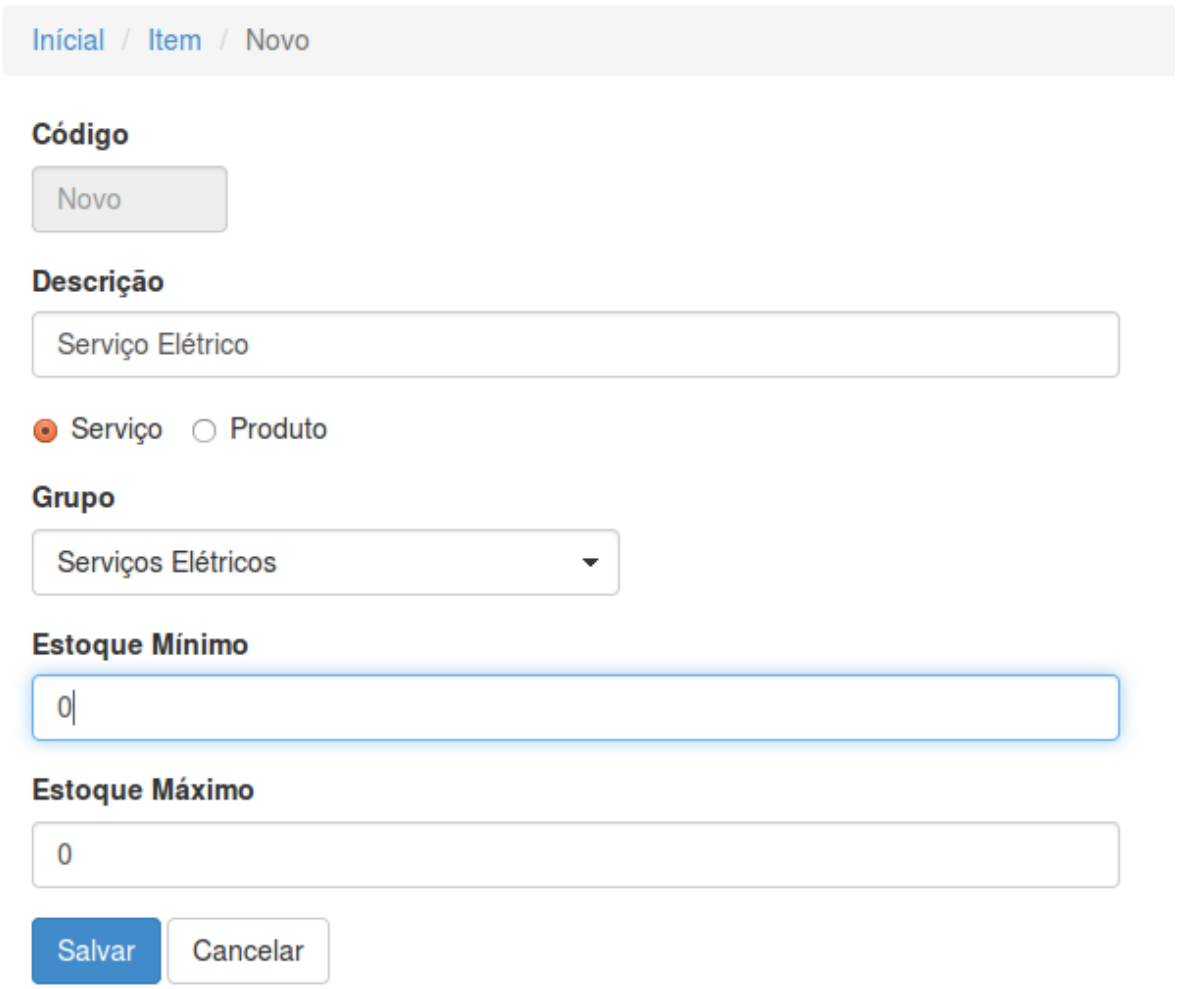

Figura 4 – Cadastro de itens

No cadastro de máquinas informa-se as máquinas presentes na indústria. Esse cadastro é necessário para vincular a OS à máquina que está recebendo a manutenção. Para lançar uma OS é necessário ter o funcionário que executou o serviço cadastrado. Para cada item de serviço na OS é necessário informar um técnico.

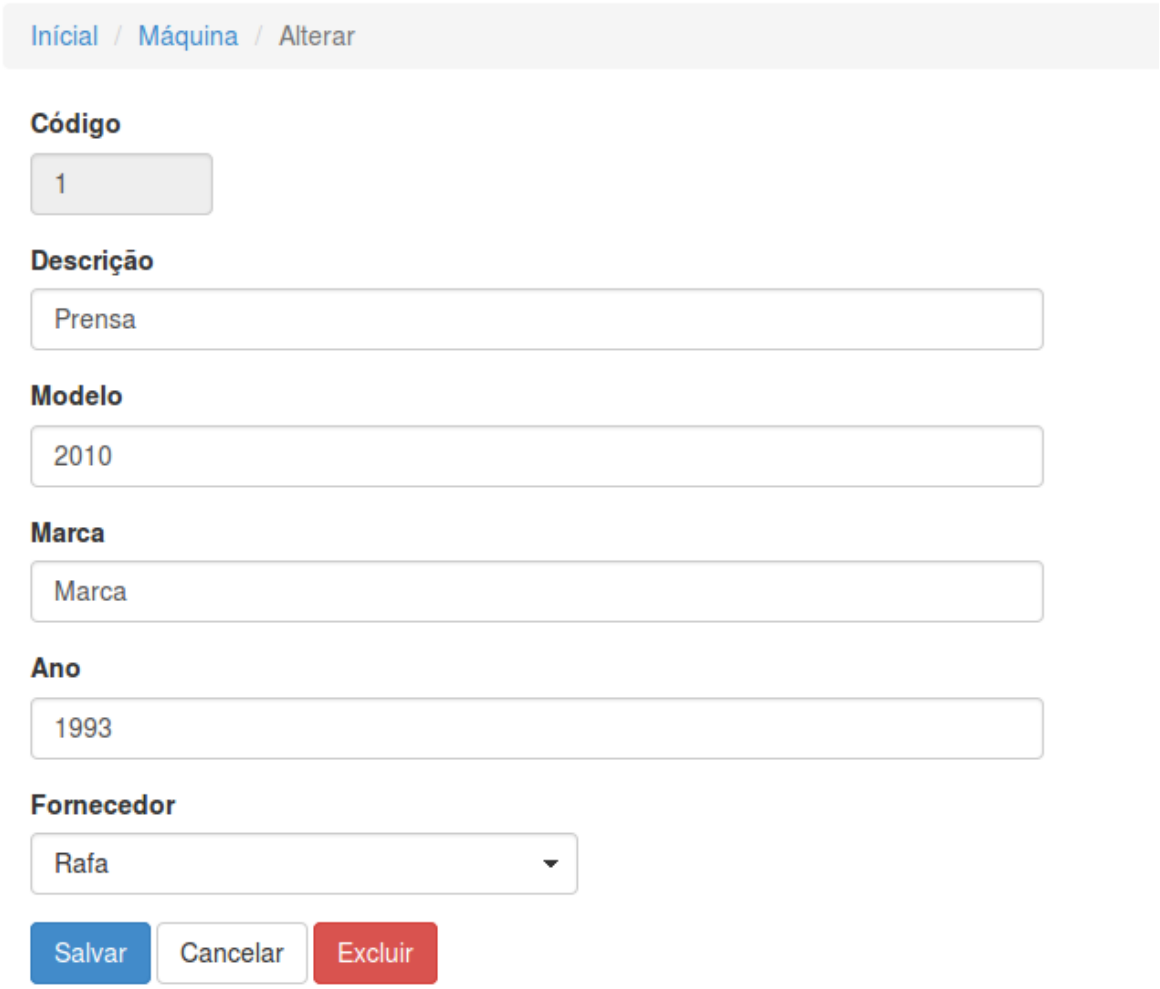

Figura 5 – Cadastro de ordens de serviço

Todos os materiais e serviços realizados são lançados na tela apresentada na Figura 6. Essa é a tela principal porque ela origina os relatórios e a tela de custos. A tela de ordem de serviço não apresenta dados relacionados aos valores. Essas informações são apresentadas em telas específicas.

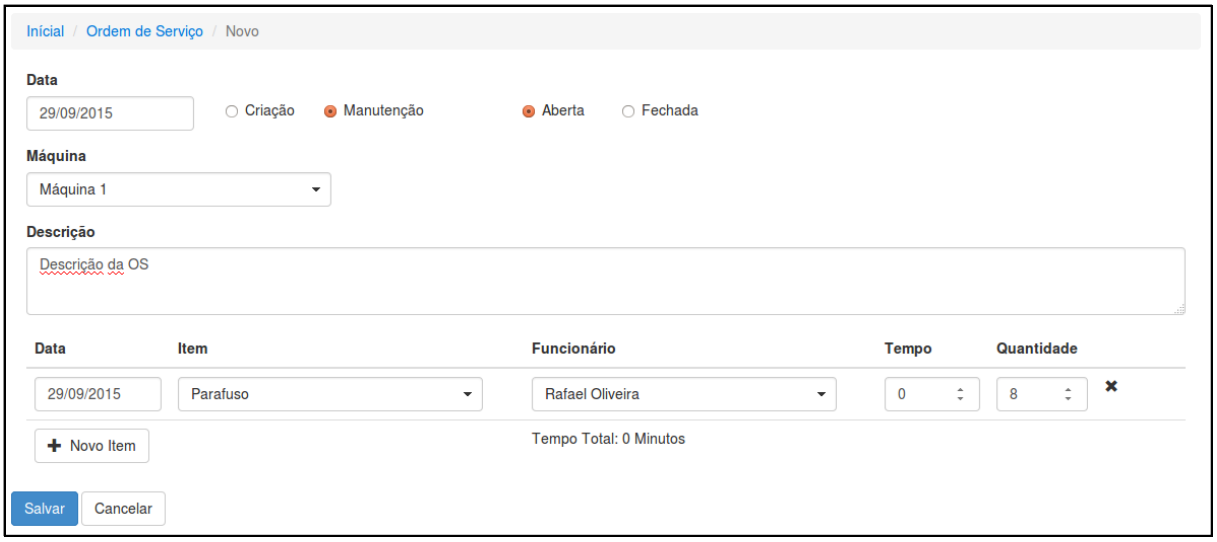

Figura 6 – Tela de ordem de serviço

A Figura 7 apresenta a tela de consulta de serviços realizados.

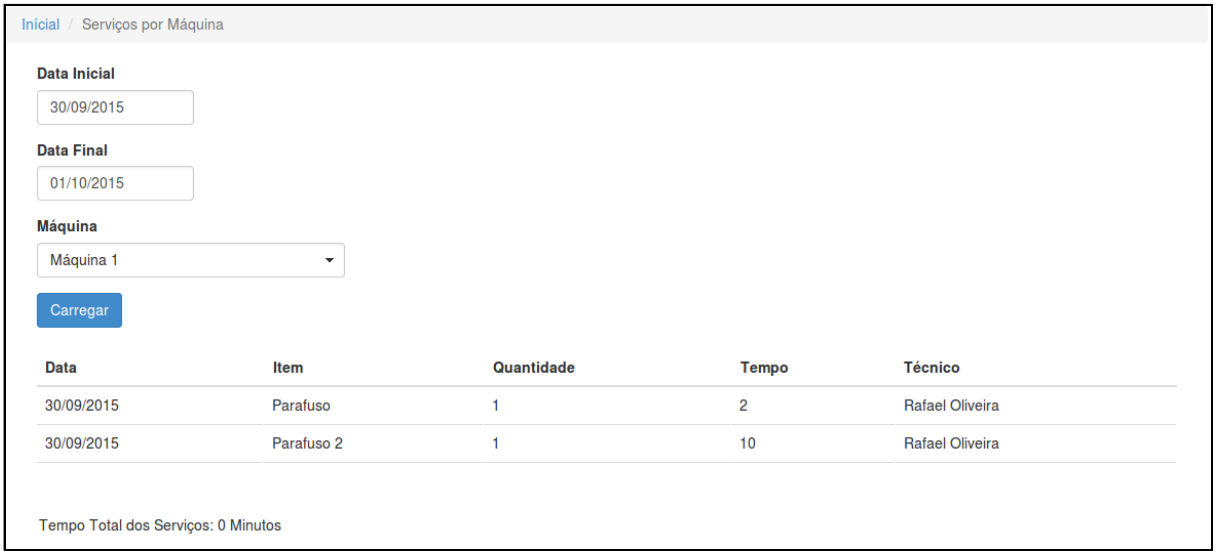

Figura 7 – Tela de consulta aos serviços realizados

A Figura 8 apresenta uma tela de cadastro que foi desenvolvida utilizando a ideia de um cadastro padrão. Os campos obrigatórios são validados quando o usuário clicar em salvar, a validação é a padrão dos componentes HTML5. Os valores de estoque mínimo e máximo são setados manualmente e tem objetivo de alertar o usuário futuramente quando houver essa funcionalidade.

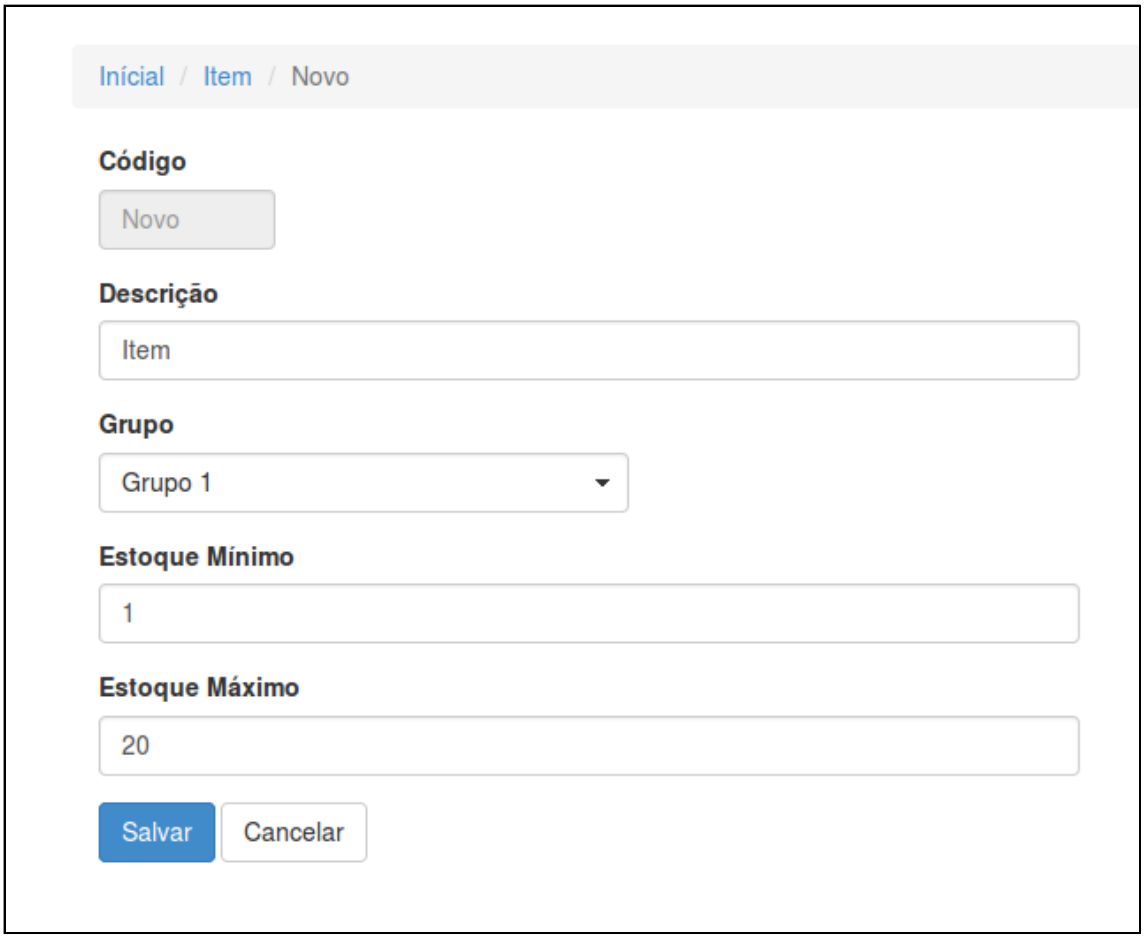

Figura 8 – Tela de cadastro de item

4.4 IMPLEMENTAÇÃO DO SISTEMA

a) Implementação do servidor

O servidor é implementado em Java e utiliza um conjunto de *frameworks* para facilitar a implementação. Assim, o foco de realização do trabalho fica direcionado para a regra de negócio. Para organizar e gerenciar todas as dependências do projeto é utilizado o Maven. O Maven baixa e cria toda a estrutura do projeto e adiciona um repositório na própria máquina com as bibliotecas informadas no pom.xml. Assim, não é mais necessário baixar manualmente cada biblioteca e importar o .jar no projeto. A Listagem 1 apresenta um exemplo de arquivo pom.xml.

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>br.edu.utfpr</groupId>
<artifactId>industria</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>war</packaging>
<parent>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-parent</artifactId>
<version>1.2.3.RELEASE</version>
</parent>
<dependencies>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
```

Listagem 1 – *Exemplo pom.xml*

Na maioria dos projetos utilizando o *Spring* do lado servidor é necessário alterar vários arquivos de configuração. No desenvolvimento do projeto foi utilizado o Spring Boot que simplifica o processo de configuração e já dispõe um servidor Tomcat embarcado para subir a aplicação. Um exemplo de configuração para setar a pasta base para os arquivos HTML e conexão com o banco é apresentado na Listagem 2.

```
spring.view.prefix: /view/
spring.view.suffix: .html
spring.jpa.hibernate.ddl-auto:update
spring.jpa.show-sql: true
spring.jpa.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
spring.datasource.url=jdbc:postgresql://localhost:5432/industria
spring.datasource.username=postgres
spring.datasource.password=cdi1
spring.datasource.driverClassName=org.postgresql.Driver
```

Listagem 2 – Exemplo configuração

Para a persistência dos dados é utilizado o *framework* objeto-realcional Spring Data, que já disponibiliza várias operações de *Create, Read, Update and Delete* (CRUD) para vários tipos de banco de dados relacionais e noSql. Internamente, o *framework* trabalha com Hibernate e *Java Persistence API* (JPA).

b) Classe modelo

A classe modelo (*model*), apresentada na Listagem 3, contém os atributos de cada entidade e mapeia a persistência do objeto.

```
@Entity
@Table(name="itens")
public class Item {
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;
@NotEmpty
@Column(length = 100, nullable = false)
private String descricao;
@NotEmpty
@Column(name="tipo", length=1, nullable=false)
private String tipo;
@ManyToOne
private GrupoItem grupo;
@Column(nullable=false)
private float estoqueMinimo;
@Column(nullable=false)
private float estoqueMaximo;
// getters e setters
```

Listagem 3 – Classe modelo

c) Interface repositório

A interface repositório disponibiliza operações de CRUD genérico e recursos adicionais de paginação de dados. A Listagem 4 apresenta a interface repositório.

```
@NoRepositoryBean
public interface GenericRepository <T> extends PagingAndSortingRepository<T, Long> {
Page<T> findTable(Pageable pageable ,String term);
List<T> findSelect2(Pageable pageable, String term);
}
```

Listagem 4 – Inteface repositório

d) Classe service

A classe *service*, código apresentado na Listagem 5, contém a implementação da regra de negócio necessária para as operações do sistema.

```
public class GenericService <T> {
private GenericRepository<T> repository;
public GenericService(GenericRepository<T> repository) {
this.repository = repository;
}
public void save(T object, HttpServletRequest request) throws Exception{
repository.save(object);
}
public void delete(T object, HttpServletRequest request) throws Exception{
repository.delete(object);
}
public void delete(Long oid, HttpServletRequest request) throws Exception{
repository.delete(oid);
}
public T findOne(Long oid) {
return repository.findOne(oid);
}
// mais funções
```

Listagem 5 – Classe service

e) Classe controller

A classe *controller* realiza a comunicação com a *view*, disponibilizando o conteúdo para cada requisição da *view* e encaminha para a *service*. A Listagem 6 apresenta código da classe *view*.

```
public abstract class GenericController <T> {
protected final Class<T> classBean;
protected final GenericService<T> service;
private final String pathList;
private final String pathCrud;
public GenericController(GenericService<T> service,Class<T> classBean, String pathList,String pathCrud) {
this.service = service;
this.classBean = classBean;
this.pathList = pathList;
this.pathCrud = pathCrud;
}
@RequestMapping( method = RequestMethod.GET)
```

```
public String getViewList() {
return pathList;
}

@RequestMapping(value = {"/novo", "/alterar/{id}"}, method = RequestMethod.GET)
public String getViewCrud(){
return pathCrud;
}

@RequestMapping("/getObjectData/{id}")
public @ResponseBody T getObjectData(@PathVariable Long id) {
if (id > 0) {
return service.findOne(id);
} else {
return getInstanceOfObjectT();
}
}

@RequestMapping(value = { "/salvar" }, method = RequestMethod.POST)
public @ResponseBody ValidationResponse save(@Valid @RequestBody T object, BindingResult result,
HttpServletRequest request) throws Exception {
ValidationResponse res = new ValidationResponse();

if (result.hasErrors()){
res.setStatus("400");
res.addAllMessageError(result.getAllErrors());
} else {
try {
service.save(object, request);
res.setStatus("200");
}catch (Exception e) {
res.setStatus("400");
res.addMessageError(ExceptionFilter.filterMessage2(e, request));
}
}
return res;
}
```

Listagem 6 – Classe *controller*

Admitindo que essas classes são necessárias para o funcionamento dos recursos do sistema e é uma boa prática o reuso de código. A padronização com classes genéricas de cadastros e listagens é viável e essas padronizações foram apresentadas nas Listagens anteriores.

f) Permissões

Para a definição de permissões de acesso foi utilizado o *framework* Spring Security. A cada *Uniform Resource Identifier* (URL) solicitada é verificado se o usuário tem permissão para acesso da mesma, se não tiver retorna um erro 403 e se o usuário ainda não foi autenticado é retornada a página de *login*. 403 é um código de erro HTTP que é retornado pelo servidor *web* quando o usuário ou aplicativo tenta acessar um recurso que aquele usuário ou aplicativo não possui acesso. É um código de erro de permissão de acesso.

g) **Camada de apresentação**

A camada de apresentação requisita dados do servidor e proporciona a interação com o operador do sistema. Para a implementação da camada de apresentação foram utilizados alguns *frameworks frontend.* O servidor devolve e recebe dados no formato *JavaScript Object Notation* (JSON) usando HTML somente na requisição das *views*, isso diminui o tráfego de dados entre cliente e servidor e se necessário é possível utilizar outra linguagem para comunicação com o servidor.

Para trabalhar com os dados em formato JSON do lado cliente foi utilizado o AngularJS que é um *framework* desenvolvido pela Google. Esse *framework* facilita a implementação das regras de negócio na camada de apresentação.

O arquivo JavaScript contém a instancia da aplicação AngularJS e a implementação das regras de negócio para buscar, salvar, editar, mensagens e etc.

Outro recurso do AngularJS é a *singlepage* cuja definição é criar uma página com o HTML base que nada mais é que a estrutura de todas as páginas. Nessa página, o menu é adicionado o menu e são declarados todos os arquivos *Cascading Style Sheets* (CSS) e JSON que serão utilizados. Esses arquivos são carregados somente uma vez e mantidos em cache, tornando, assim, a navegação pelas páginas do sistema bem mais rápida. A Listagem 7 apresenta o HTML da página *index*.

```
<!DOCTYPE html>
```

```
<html lang="en" ng-app="App">
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Controle de Manutenção</title>
<!-- Declaração do CSS -->
</head>
<body>
<!-- HTML fixo Ex: menu da aplicação -->
<div ng-view>
<!-- HTML das páginas são carregados aqui -->
</div>
<!-- Declaração do JS -->
</body>
</html>
```

Listagem 7 – Index.html

Um exemplo de página de listagem é apresentado na Listagem 8.

```
<div class="container" style="padding: 30px 15px 0;">
<div id="content">
<breadcrumbs label="Grupos"> </breadcrumbs>
<a class="btn btn-primary" ng-click="getViewNovo()">Adicionar Novo</a>
<br>
<br>
<table id="table" ng-init="initTable()" data-toggle="table" data-classes="table table-no-bordered"
data-height="600" data-side-pagination="server" data-page-list="[5, 10, 20, 50, 100, 200]" data-
pagination="true">
<thead>
<tr>
<th data-field="nome" data-align="left">Nome</th>
</tr>
</thead>
</table>
</div>
</div>
```

Listagem 8 – Exemplo de página de listagem

Um exemplo de página de inserção e alteração é apresentado na Listagem 9.

```
<div class="container" style="padding: 30px 15px 0;" ng-init="getData()" >
<breadcrumbs label="Grupo"> </breadcrumbs>
<div class="row">
<div class="col-xs-6">
<form name="form">
<input type="hidden" name="id" id="id" ng-model="row.id" />
<div class="form-group">
<label class="control-label" for="name">Nome</label>
<input class="form-control" type="text" name="nome" id="nome" ng-model="row.nome" placeholder="Nome"
required="required" />
</div>
<div btn-salvar-cancelar-excluir></div>
</form>
</div>
</div>
</div>
```

Listagem 9 – Exemplo de página de inserção e alteração

Com o uso desses recursos e tecnologias, o código de cada página fica mais simples. O desempenho da aplicação é melhorado e efetividade no carregamento das páginas na aplicação.

Outro recurso interessante que ajuda a deixar o desenvolvimento das páginas mais simples são as diretivas do AngularJS. Com elas é possível criar *templates* HTML e chamá-los dentro das páginas. Exemplo desses *templates* são os botões salvar, cancelar, excluir que estão presentes em todas as telas de cadastro. A Listagem 10 apresenta a declaração da diretiva no *controller* do AngularJS.

```
app.directive('btnSalvarCancelarExcluir', function () {
return {
template: '<input class="btn btn-primary" type="submit" ng-click="save(row)" value="Salvar" />'+
'<a class="btn btn-default" ng-click="cancelar()" style="margin-left: 3px">Cancelar</a>'+
'<a class="btn btn-danger" ng-click="deletar(row)" ng-show="row.id > 0" style="margin-left: 3px"
>Excluir</a>'
};
});
```

Listagem 10 – Declaração da diretiva no *controller* do AngularJS

A declaração da diretiva nada mais é que o retorno do código HTML que vai ser injetado na página. A transferência dos dados entre o servidor e a página é feita em formato

JSON. O processo para buscar e salvar dos cadastros foi padronizado dentro de um *controller* genérico do AngularJS, cujo código está apresentado na Listagem 11. Assim, para criar um cadastro novo, pouquíssima implementação é necessária.

```
$scope.getViewNovo = function() {
$location.path('/'+getParamUrl($routeParams, 0)+'/'+getParamUrl($routeParams, 1)+'/novo');
}
$scope.getViewEdicao = function(id) {
$location.path('/'+getParamUrl($routeParams, 0)+'/'+getParamUrl($routeParams, 1)+'/alterar/' + id);
}
$scope.cancelar = function() {
$location.path('/'+getParamUrl($routeParams, 0)+'/'+getParamUrl($routeParams, 1)+'/');
}
$scope.getData = function() {
var id = getParamUrl($routeParams, 3);
if (id == undefined) {
id = 0;
}
$http.get('/cadastro/'+getParamUrl($routeParams, 1)+'/getObjectData/'+id).success(function(Obj){
$scope.row = Obj;
$scope.$broadcast('afterGetData');
if($scope.row.itens == null){
$scope.row.itens = [];
$scope.row.itens.push({});
}
};
}
```

Listagem 11 – *Controller* genérico

No *controller* genérico apresentado na Listagem 12 foi feito uso do recurso routeParams que contém toda a URL que a página está. Essa variável é alimentada na rota da aplicação AngularJS no qual todas as URLs digitadas passam e o *controller* genérico é vinculado com todas as páginas.

```
app.config(function($routeProvider){
// cadastros normais
$routeProvider.when('/:recurso*',{
templateUrl: function(urlattr){
console.log(urlattr);
return '/' + urlattr.recurso;
},
controller : 'GenericCtrl'
});
});
```

Listagem 12 – Função routeProvider

5 CONCLUSÃO

O aplicativo desenvolvido tem como objetivo principal auxiliar no gerenciamento de ordens de manutenção de máquinas industriais. Para isso os requisitos foram levantados a partir das necessidades e interesses de uma indústria da área de metalurgia. Assim, o sistema visa atender os requisitos identificados a partir de um fluxo de processo que foi ajustado com o desenvolvimento do sistema, mais especificamente da fase de levantamento de requisitos.

O referencial teórico abordou conceitos relacionados à manutenção de máquinas industriais por ser esse o escopo e contexto do aplicativo desenvolvido. As tecnologias utilizadas apresentaram recursos que facilitaram o desenvolvimento pela facilidade de padronizar o *front-end*, assim todos os cadastros utilizam o mesmo código.

Inicialmente foi utilizada a tecnologia JavaServer Pages para o desenvolvimento. Muitas dificuldades foram encontradas para trabalhar com o *front-end* principalmente em relação à necessidade de campos dinâmicos em formulário, sem atualizar a página.

Para o desenvolvimento do aplicativo foi criado um padrão para se implementar os cadastros, que geralmente são muitos em um projeto e tomam bastante tempo dos desenvolvedores. Esse padrão permite o reuso de funcionalidades e agiliza o desenvolvimento, permitindo ao programador centrar-se nas funcionalidades e regras de negócio.

Como resultado deste trabalho, uma solução simples de controle de manutenção para uma indústria foi implementada. Contudo, o desenvolvimento de um aplicativo *web* propicia uma forma de trabalhar que ganha cada vez mais espaço por ser dinâmica e necessitando de menos transferência de dados pela Internet.

Como trabalhos futuros, complementares ao desenvolvido realizado, está o acréscimo de funcionalidades para integração com outros setores da empresa como o de estoque e financeiro.

REFERÊNCIAS

CAPETTI, Edson José. **O papel da gestão da manutenção no desenvolvimento da estratégia de manufatura.** Dissertação (mestrado). Programa de Pós-Graduação em Engenharia de Produção e Sistemas da Pontifícia Universidade Católica do Paraná, 2005.

KRUCHTEN, Philippe. **Introdução ao RUP:** rational unified process. 2 ed. Rio de Janeiro: Ciência Moderna, 2004.

PINTO, Alan Kardec; XAVIER, Júlio A. Nascif. **Manutenção: função estratégica**. 2 ed. Rio de Janeiro: Qualitymark, 2003.

PRESSMAN, Roger. **Engenharia de software**. 6 ed. Rio de Janeiro: McGraw-Hill, 2006.

SILVA, Romeu Paulo. **Gerenciamento do setor de manutenção.** Trabalho de conclusão de curso (Especialização) Especialização em Gestão Industrial. Universidade de Taubaté, 2004.

SILVA, Diogo Anselmini da; ANTUNES, Marcos Vinicius. **Proposta de implantação da manutenção preventiva em um supermercado do Oeste do Paraná**. Monografia de conclusão de curso. Curso Superior de Tecnologia em Manutenção Industrial da Universidade Tecnológica Federal do Paraná – UTFPR, 2012.

SLACK, Nigel; CHAMBERS, Stuart; JOHNSTON, Robert. **Administração da produção**. 2 ed. São Paulo: Atlas, 2002.

XAVIER, Julio N. **Manutenção: tipos e tendências**. TECEM. Disponível em: <http://tecem.com.br/site/downloads/artigos/tendencia.pdf>. Acesso em: 20 abr. 2015.