UNIVERSIDADE ESTADUAL DE MARINGÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE INFORMÁTICA

ENGENHARIA DE SOFTWARE II

|  |
| --- |
| **RELATÓRIO DE PROJETO**  **“Sistema Pastelaria LM”**  Gilberto Leal Santos 99999  Luiz Gustavo Riberio 68651  Tiago Uemura 94108  **2018** |

MARINGÁ

Sumário

[1. Introdução 4](#_Toc530331981)

[2. Revisão Literária 4](#_Toc530331982)

[2.1. Gerenciamento de Configuração de Software 4](#_Toc530331983)

[2.1.1. Gerenciamento de Mudanças (GM) 4](#_Toc530331984)

[2.1.2. Gerenciamento de Versões (GV) 5](#_Toc530331985)

[3. Resultados 5](#_Toc530331986)

[4. Conclusão 5](#_Toc530331987)

[5. Bibliografia 6](#_Toc530331988)

**TO DO =======================================================================**

**COD**

**05/12/20218 13:30**

**Controlador.cadastrarProduto(): retirei a função de add dele e coloquei a responsabilidade para ProdutoDao.adicionarProduto() ------desta forma não utiliza o controlador.buscarProduto()**

**Construtor de Pedido deve ser sem parâmetros, para ser criado vazio e colocado os produtos em sua lista e depois as outras informações. Controlador.criarPedido(): criei um sem parâmetro e comentei que estava com parâmetros. pedidoDao.criarPedido() e Implements mesma coisa**

**// Pedido pd = new Pedido(); linha 61 InterfaceAtendente**

**Não compete a interface criar, coloquei no controlador.criarPedido() que chama o DAO (talvez o controlador mesmo possa criar), então na interface atendente chama o método do controlador.**

**\*\* ainda não feito**

**\*\*Criar método public void menuTipoPedido(): mostra os enums de tipo pedido e pede para selecionar um.**

**\*\*: Colocar novoPedido() no controlador**

**Produto produto = new Produto();**

**interfaceAtendente.menuTipoPedido(); //criar instancia da interface**

**case 1, 2,**

**3: while etc**

**Controlador.incluirPedido(): pega o produto p e setQuantidade, depois pede para Dao colocar o produto em pedido pd**

**fecharPedido(): vai incluir as outras informações no pedido. calcula o valor total em controlador**

**imprimirPedido(): vai imprimir o pedido pelo toString da classe Pedido;**

**Por que não listar os produtos já cadastrados e pedir pro usuário selecionar entre 1 a n os produtos? (utilizando case): para evitar erro no pedido, o user pode apertar o número errado. É melhor escrever o nome do produto.**

TO DO

DOING

DONE

Dito pelo pdf da Juliana

*Dito por Gil, Luiz ou Tiago*

**3) O que deve estar contido no relatório do trabalho:**

a) Identificação da equipe.

b) Descrição dos passos necessários para a execução do projeto.

c) Pesquisa sobre gerenciamento de configuração, citando fontes científicas e confiáveis, apresentando no final do relatório as referências bibliográficas conforme o padrão ABNT.

1. *Gerenciamento de mudanças*
2. *Gerenciamento de versões*
3. *Construção do sistema*
4. *Gerenciamento de release*

d) Com base na pesquisa, deve ser apresentada uma descrição do histórico de gerenciamento de configuração realizada durante o desenvolvimento do trabalho prático.

e) Pesquisa sobre técnicas e estratégias de teste de software, citando fontes científicas e confiáveis, apresentando no final do relatório as referências bibliográficas conforme o padrão ABNT.

f) Com base na pesquisa, deve ser apresentada uma forma de implementação de teste de software para o projeto desenvolvido.

g) Conclusão:

• Destaque o que foi implementado;

• Descreva como as tarefas foram atribuídas aos membros da equipe;

• Explique como as tarefas foram gerenciadas;

• Discuta sobre a utilidade dos testes de software e como foram implementados na prática.

• Descreva as dificuldades encontradas.

============================================================

1. Introdução

Descrição dos passos necessários para a execução do projeto.

1. Revisão Literária
   1. Gerenciamento de Configuração de Software

Sistemas de softwares sofrem alterações a partir do início do seu desenvolvimento, durante seu uso, passam por novas versões, e assim segue até o dia que não seja mais utilizado ou aproveitável. Essas mudanças geram grande conflitos caso não sejam bem gerenciadas, exemplos são: alterações nos requisitos organizacionais, reparos de bugs, alterações do ambiente, seja software ou hardware.

Segundo Sommerville (2011), o Gerenciamento de Configuração de Software (GCS) é o conjunto de políticas, processos e ferramentas que é utilizado no gerenciamento de mudanças de sistemas de software.

O autor define quatro atividades do GCS:

1. Gerenciamento de mudanças
2. Gerenciamento de versões
3. Construção do sistema
4. Gerenciamento de releases

Entende-se que efetuar essas quatros atividades auxilia a uma equipe ou o gerente do projeto a ter o controle de quais mudanças foram feitas em quais versões de cada componente de software, mesmo que estejam sendo desenvolvidas em paralelo com vários desenvolvedores em lugares remotos. O GCS melhora o fluxo de trabalho, minimiza o retrabalho pois a informação está acessível a todos e diminui o risco de perda de dados.

A seguir apresenta-se uma breve definição de cada atividade dentro do GSC, segundo Sommerville (2011).

* + 1. Gerenciamento de Mudanças (GM)

O objetivo desta atividade é garantir a evolução do sistema seja um processo controlado, por alterações prioritárias e efetivas, no momento certo e do jeito certo.

O GM implica em analisar os custos e benefícios de cada alteração proposta, aprovar o desenvolvimento daquelas que valem a pena no momento e acompanhar os resultados das alterações.

Este processo se inicia pela requisição de mudança (CR, Change Request) no sistema por algum cliente. Geralmente, utiliza-se um formulário para realizar a requisição (CRF, Change Request Form), assim as informações da CR podem ser compartilhadas entre todos envolvidos, e à medida que é processada pode-se incluir registros sobre as decisões em cada etapa. O CRF pode conter, além da mudança, recomendações, estimativas de custos, datas de início, aprovação, conclusão etc.

A aprovação da CR deve considerar alguns fatores antes de ser passado para a equipe de desenvolvimento como, por exemplo, as consequências de não fazer a mudança, os benefícios, os custos, o ciclo de release (entregas de versões ao cliente), etc.

Quando a equipe de desenvolvimento altera um componente de software, é necessário registrar esta mudança em um histórico de derivação, este pode estar no cabeçalho do código fonte em forma de comentário, contendo a referência ao CR, autor da modificação, data, razão etc.

Muitos processos são realizados no tratamento de CR e por isso algumas ferramentas são desenvolvidas para automatizar desde a etapa de proposta inicial até a aprovação da mudança.

* + 1. Gerenciamento de Versões (GV)

Os itens de configuração de um software podem ter várias versões independentes devido a necessidade de alterações. Cada item tem seu conjunto de versões chamados de *codelines*. O sistema de software pode ter várias configurações distintas e independentes, combinando as versões de cada item de configuração. Este conjunto de componentes definido para o sistema é chamado de *baseline*, que podem ser especificadas de acordo com uma referência a cada componente e sua versão, por exemplo, X.1.2.

O GV é o processo que gerencia os *codelines* e *baselines*, ou seja, identificar, armazenar, controlar o acesso as versões de componentes. O GV garante que desenvolvedores possam alterar versões dos componentes sem interferir umas às outras, permite recriar versões de sistemas completos para outro cliente, e com auxílio de ferramentas específicas podem automatizar alguns processos como a identificação de versão e release, gerenciamento de armazenamento, registro de histórico de mudanças, desenvolvimento independente, suporte a projetos etc.

Para permitir o trabalho simultâneo em um mesmo componente do sistema, sistemas de GV utiliza-se um repositório público e espaço de trabalho privado para realizar o *check*-*out* de componente do repositório público para trabalho no espaço privado, e quando concluir as alterações faz o *check-in* do componente para o repositório. Desta forma, quando uma pessoa tentar realizar o *check*-*out* de um componente já realizado por outra, o sistema deve avisar o usuário.

Com o desenvolvimento independente assegurado pode-se surgir ramificações dos *codelines* que são diferentes continuações de uma versão anterior em comum. As ramificações criadas podem se fundir em algum momento para criar uma nova versão.

* + 1. Construção de sistemas

É o processo de ligar os componentes de sistemas criados separadamente para criar um sistema completo e executável. Podem ter três plataformas de sistemas envolvidos e cada um está sujeito a erros em sua construção

1. O sistema de desenvolvimento (compiladores, editores código-fonte etc.)
2. Servidor de construção (construção de versões definitivas e executáveis)
3. Ambiente alvo (plataforma de execução)

Pesquisa sobre técnicas e estratégias de teste de software

1. Resultados

Com base na pesquisa, deve ser apresentada uma descrição do histórico de gerenciamento de configuração realizada durante o desenvolvimento do trabalho prático.

Com base na pesquisa, deve ser apresentada uma forma de implementação de teste de software para o projeto desenvolvido.

1. Conclusão

Destaque o que foi implementado;

• Descreva como as tarefas foram atribuídas aos membros da equipe;

• Explique como as tarefas foram gerenciadas;

• Discuta sobre a utilidade dos testes de software e como foram implementados na prática.

• Descreva as dificuldades encontradas.

1. Bibliografia

[1] SOMMERVILLE, I. Engenharia de Software. 9 ed. São Paulo: Pearson Addison Wesley, 2011. .