Murilo César Ferreira

Análise e Desenvolvimento de Sistemas – UniSenai

Gerência de Configuração e Manutenção de Software

O que é

A Gerência de Configuração e Manutenção de Software é um conjunto de conceitos, boas práticas, ferramentas e processos essenciais na Engenharia de Software para garantir que um sistema ou produto de software seja desenvolvido, mantido e evoluído de forma controlada e eficiente. A Gerência de Configuração (GCS) envolve a identificação, organização e controle de itens de configuração, como código-fonte, documentação e bibliotecas, utilizando ferramentas como Git, SVN e Mercurial. Essas ferramentas ajudam a implementar boas práticas, como o versionamento de código, a revisão de mudanças e a garantia de rastreabilidade. Além disso, processos como o controle de mudanças e a auditoria de configuração são fundamentais para evitar inconsistências e garantir que todos os envolvidos no projeto trabalhem com a mesma versão dos artefatos.

Já a Manutenção de Software abrange processos como correção de defeitos (manutenção corretiva), adaptação a novos ambientes (adaptativa), melhorias de desempenho (perfeitiva) e prevenção de problemas futuros (preventiva). Para isso, é essencial adotar boas práticas, como a documentação clara das alterações, a realização de testes regulares e a priorização de mudanças com base no impacto. Ferramentas de gestão de projetos, como Jira e Trello, e de integração contínua, como Jenkins, são amplamente utilizadas para apoiar esses processos. A integração entre Gerência de Configuração e Manutenção de Software é crucial, pois a primeira fornece a base para que as atividades de manutenção sejam realizadas de forma organizada e segura.

Por todos esses aspectos, fica claro que a Gerência de Configuração e Manutenção de Software são áreas interligadas que dependem de conceitos claros, boas práticas bem definidas, ferramentas adequadas e processos estruturados. Ao adotar essas práticas, as organizações podem garantir a qualidade, a eficiência e a rastreabilidade de seus sistemas, reduzindo custos e riscos associados a mudanças não controladas. Isso permite que o software evolua de forma consistente, atendendo às necessidades dos usuários e se adaptando às mudanças tecnológicas e de negócios.

Plano de Manutenção - TechFood

Descrição do Sistema

O sistema de pedidos do Restaurante TechFood é uma aplicação web interna, acessada exclusivamente pela equipe do restaurante. Ele foi projetado para simplificar o gerenciamento de pedidos, comandas e relatórios, atendendo às necessidades dos garçons, cozinheiros e caixa.

O garçom acessa o sistema por meio de um celular, utilizando um navegador web. Sua principal função é abrir a conta da mesa e anotar os pedidos dos clientes diretamente no sistema.

Na cozinha, os pedidos são recebidos em um computador, onde são gerenciados por ordem de chegada. O sistema permite que as comandas sejam impressas automaticamente, facilitando o trabalho do cozinheiro e garantindo que os pedidos sejam preparados na sequência correta.

No caixa, o atendente é responsável por fechar a conta da mesa quando o cliente finaliza o pedido. Além disso, o sistema gera relatórios de faturamento diário e mensal, bem como controla o gasto de insumos com base nos pedidos realizados, ajudando na gestão financeira e operacional do restaurante.

Em termos de tecnologias, o sistema utiliza Vue.js com Bootstrap no frontend, garantindo uma interface simples e responsiva que funciona bem tanto em celulares quanto em computadores. No backend, a aplicação é desenvolvida em C#/.NET, com APIs RESTful para processar os pedidos e integrar com o banco de dados. O banco de dados utilizado é o PostgreSQL, que armazena todas as informações necessárias, como pedidos, cardápio, clientes e funcionários.

Após a implantação do sistema, alguns problemas surgiram, demonstrando a importância do plano de manutenção preventiva. Em um dos incidentes, durante o horário de pico, os pedidos começaram a demorar para serem processados, e o sistema apresentou lentidão generalizada. A equipe técnica identificou que a CPU do servidor estava operando acima de 90% de uso, impactando diretamente no tempo de resposta das APIs. Esse problema causou atrasos na cozinha, uma vez que os pedidos não apareciam imediatamente na tela para impressão, e os garçons enfrentaram dificuldades para registrar novas comandas.

Outro problema ocorreu quando uma atualização não testada foi aplicada diretamente em produção, resultando em falhas inesperadas na exibição das comandas. Durante algumas horas, a cozinha não recebeu os pedidos corretamente, forçando os garçons a anotarem as comandas manualmente, o que gerou confusão e erros na entrega dos pratos. A falta de um ambiente de homologação adequado para testar essas atualizações foi um fator crítico nesse incidente.

Além disso, um incidente de segurança ocorreu quando um usuário não autorizado tentou explorar vulnerabilidades no sistema, tentando acessar informações restritas. Embora o ataque não tenha sido bem-sucedido, foi identificado que a comunicação entre o frontend e o backend não estava sendo forçada via HTTPS, tornando o tráfego de dados suscetível a interceptações. Esse problema revelou a necessidade de reforçar as práticas de segurança para proteger o sistema contra acessos indevidos.

Felizmente, todos esses problemas já haviam sido previstos no plano de manutenção preventiva. O monitoramento contínuo com Prometheus e Grafana teria detectado a sobrecarga da CPU antes que afetasse o desempenho do sistema, acionando alertas para que a equipe tomasse medidas corretivas rapidamente. A exigência de testes em ambiente de homologação antes de atualizações em produção teria evitado a falha na exibição das comandas, garantindo que todas as mudanças fossem verificadas previamente. Já a realização de varreduras de segurança trimestrais com OWASP ZAP e a obrigatoriedade

do uso de HTTPS teriam impedido a vulnerabilidade na comunicação de dados, protegendo o sistema contra acessos não autorizados. Esses eventos destacam como a aplicação rigorosa do plano de manutenção preventiva pode garantir um funcionamento mais estável e seguro para o sistema do Restaurante TechFood.

1. Monitoramento de Desempenho

O objetivo do monitoramento de desempenho é garantir que o sistema funcione sem lentidão ou falhas durante o horário de pico. Para isso, serão utilizadas ferramentas como Prometheus e Grafana para monitorar o uso de CPU e memória do servidor, o tempo de resposta das APIs e o número de pedidos processados por minuto. Alertas serão configurados para notificar a equipe em caso de uso de CPU acima de 70%, tempo de resposta das APIs acima de 1 segundo ou erros HTTP (ex.: 5xx) acima de 1% das requisições.

2. Atualizações Regulares

O objetivo das atualizações regulares é manter o sistema seguro e estável. As dependências do projeto, como Vue.js, Bootstrap e .NET, serão atualizadas trimestralmente. Antes de aplicar as atualizações em produção, elas serão testadas em um ambiente de homologação. Além disso, o sistema operacional do servidor EC2 será mantido atualizado com os patches de segurança mais recentes.

3. Testes Automatizados

O objetivo dos testes automatizados é garantir que o sistema funcione corretamente após alterações. Serão implementados testes unitários para funções críticas no backend, como cálculo de totais e geração de relatórios, utilizando ferramentas como xUnit ou NUnit. Testes de integração garantirão que as APIs funcionem corretamente, enquanto ferramentas como Jest ou Cypress serão usadas para testes no frontend. Todos os testes serão integrados em um pipeline de CI/CD usando GitHub Actions.

4. Backup de Dados

O objetivo do backup de dados é prevenir perda de informações em caso de falhas. Backups diários completos e incrementais a cada 6 horas serão realizados e armazenados na AWS S3, com versionamento habilitado. Mensalmente, a restauração dos backups será testada para garantir que funcionem corretamente.

5. Revisão de Código e Documentação

O objetivo da revisão de código e documentação é manter o código limpo e a documentação atualizada. Todas as alterações no sistema passarão por revisões de código (code reviews). As mudanças serão documentadas em um changelog.md, e um README.md será mantido atualizado com instruções para configurar e rodar o sistema. As APIs serão documentadas usando Swagger ou Postman.

6. Plano de Resposta a Incidentes

O objetivo do plano de resposta a incidentes é garantir uma resposta rápida e eficiente em caso de falhas. Um protocolo será criado para lidar com incidentes, incluindo a identificação do problema, notificação da equipe responsável, isolamento da causa raiz, aplicação da solução e documentação do incidente. Um documento incident_response.md será criado com soluções para problemas comuns, como banco de dados offline, APIs lentas ou indisponíveis e erros na impressão de comandas.

7. Segurança

O objetivo da segurança é proteger o sistema contra acessos não autorizados e vulnerabilidades. Será implementada autenticação básica (usuário e senha) para acessar o sistema, e as entradas de usuário serão validadas para prevenir injeção de SQL e XSS. A comunicação entre o frontend e o backend será protegida com HTTPS, e varreduras de segurança trimestrais serão realizadas usando ferramentas como OWASP ZAP.

8. Melhorias Contínuas

O objetivo das melhorias contínuas é aperfeiçoar o sistema com base no feedback da equipe. O feedback dos garçons, cozinheiros e caixa será coletado regularmente, e as métricas de desempenho serão analisadas para identificar oportunidades de melhoria. Novas funcionalidades serão planejadas e implementadas conforme as necessidades do restaurante

Ferramentas Utilizadas

Para garantir a qualidade, segurança e eficiência do projeto, diversas ferramentas foram selecionadas para atender às necessidades específicas de cada função. O Prometheus e o Grafana são utilizados para monitoramento, permitindo a coleta e visualização de métricas em tempo real, auxiliando a equipe a identificar problemas rapidamente. No fluxo de CI/CD, o Github Actions automatiza testes, builds e deploys, garantindo que novas versões sejam entregues de forma confiável e eficiente.

A integridade dos dados é assegurada pelo AWS S3, que armazena backups diários em um ambiente seguro e altamente disponível. Para garantir a estabilidade do código, o projeto utiliza xUnit/NUnit no backend e Jest/Cypress no frontend, ferramentas amplamente adotadas para testes automatizados, permitindo a detecção precoce de falhas antes da liberação de novas versões. Já a documentação da API e do sistema é organizada com Swagger, Postman e Markdown, facilitando o acesso e a compreensão dos recursos por toda a equipe.

Na área de segurança, o OWASP ZAP auxilia na identificação de vulnerabilidades, realizando varreduras automatizadas para detectar falhas antes que possam ser exploradas. Além disso, a utliização de HTTPS garante garante a criptografia dos dados trafegados, protegendo a comunicação contra ataques. ESsas ferramentas foram escolhidas por sua confiabilidade e adoção consolidada no mercado, garantindo um desenvolvimento seguro, eficiente e bem documentado.

Função	Ferramenta
Monitoramento	Prometheus, Grafana
CI/CD	GitHub Actions
Backup	AWS S3
Testes	xUnit/NUnit (backend), Jest/Cypress (frontend)
Documentação	Swagger, Postman, Markdown
Segurança	OWASP ZAP, HTTPS

Cronograma de Manutenção

Tarefa	Frequência	Responsável
Monitoramento de desempenho	Contínuo	Equipe de DevOps
Atualizações de dependências	Trimestral	Desenvolvedores
Testes automatizados	A cada nova versão	Desenvolvedores
Backup de dados	Diário	Equipe de DevOps
Revisão de código	A cada alteração	Desenvolvedores
Varredura de segurança	Trimestral	Equipe de Segurança

Conclusão

Este plano de manutenção preventiva foi criado para atender às necessidades do sistema simplificado do **Restaurante TechFood**, que é focado no uso interno pela equipe do restaurante. Ele garante que o sistema funcione de forma estável, segura e eficiente, evitando problemas que possam impactar o atendimento aos clientes e a gestão do negócio.

Referências

AWS. **O que é gerenciamento de configuração?** 2024. Disponível em: https://aws.amazon.com/pt/what-is/configuration-management/. Acesso em: 21 fev. 2025.

DEVMEDIA (Brasil). **Gerência de Configuração de Software**. 2009. Disponível em: https://www.devmedia.com.br/gerencia-de-configuracao-de-software/9145. Acesso em: 21 fev. 2025.

CHINA, Chrystal R.; GOODWIN, Michael. **O que é gerenciamento de configuração (CM)?** 2024. Disponível em: https://www.ibm.com/br-pt/topics/configuration-management. Acesso em: 21 fev. 2025.

BUCHANAN, Ian. **Gerenciamento de configurações**: como o gerenciamento de configuração ajuda as equipes de engenharia a criar sistemas robustos e estáveis. Como o gerenciamento de configuração ajuda as equipes de engenharia a criar sistemas robustos e estáveis. 2024. Disponível em: https://www.atlassian.com/br/microservices/microservices-architecture/configuration-management. Acesso em: 21 fev. 2025.S