ENGENHARIA DE SOFT

DIAGRAMAS

1. **Diagrama de Caso de Uso:**
   - Mostra as interações entre os atores externos e o sistema.
   - Ajuda a entender os diferentes casos de uso do sistema e como os usuários interagem com ele.
2. **Diagrama de Classes:**
   - Representa as classes do sistema, seus atributos e métodos, e os relacionamentos entre elas.
   - Ajuda a visualizar a estrutura do sistema e como as classes se relacionam umas com as outras.
3. **Diagrama de Sequência:**
   - Mostra a interação entre os objetos do sistema ao longo do tempo.
   - Ajuda a entender a sequência de eventos em um processo ou funcionalidade específica.
4. **Diagrama de Implantação:**
   - Descreve a arquitetura física do sistema, incluindo hardware, software, redes e outros recursos de hardware.
   - Ajuda a visualizar como o sistema será implantado em um ambiente de produção.
5. **Diagrama de Estado:**
   - Representa os diferentes estados que um objeto pode assumir durante sua vida no sistema.
   - Útil para modelar comportamentos complexos de objetos que mudam de estado com base em eventos.
6. **Diagrama Entidade-Relacionamento (ER):**
   - Mostra as entidades do sistema, seus atributos e os relacionamentos entre elas.
   - Útil para modelar a estrutura de um banco de dados relacional.
7. **Diagrama de Atividades:**
   - Descreve o fluxo de controle de um processo ou atividade no sistema.
   - Útil para modelar processos de negócios ou fluxos de trabalho complexos.

TESTES

Para garantir a qualidade e robustez do seu projeto PIM, você pode realizar uma série de testes em diferentes áreas. Aqui estão alguns exemplos de testes que você pode realizar:

1. **Testes de Unidade:**
   - Verificar se cada função ou método individual do seu código funciona conforme o esperado.

- Exemplos: Testes de funções que realizam cálculos, manipulação de dados, validação de entrada, etc.

2. **Testes de Integração:**
   - Testar a integração entre diferentes partes do sistema para garantir que elas funcionem bem juntas.
   - Exemplos: Testes que verificam a interação entre módulos do sistema, integração com APIs externas, integração entre front-end e back-end.

3. **Testes Funcionais:**
   - Verificar se as funcionalidades do sistema estão de acordo com os requisitos especificados.
   - Exemplos: Testes de fluxo de trabalho, testes de casos de uso, testes de interface do usuário.

4. **Testes de Aceitação do Usuário (UAT):**
   - Realizados pelo usuário final para garantir que o sistema atenda aos requisitos de negócios e às expectativas do cliente.
   - Exemplos: Testes de usabilidade, testes de desempenho, testes de acessibilidade.

5. **Testes de Regressão:**
   - Garantir que as alterações recentes no código não afetem as funcionalidades existentes.
   - Exemplos: Reexecutar testes de unidade, integração e funcionalidade após implementar uma nova funcionalidade ou correção de bug.

6. **Testes de Segurança:**
   - Verificar se o sistema é seguro contra ameaças como ataques de injeção SQL, XSS (Cross-Site Scripting), CSRF (Cross-Site Request Forgery), etc.
   - Exemplos: Testes de penetração, análise de vulnerabilidades, verificação de conformidade com padrões de segurança.

7. **Testes de Desempenho:**
   - Avaliar o desempenho do sistema em termos de tempo de resposta, carga máxima suportada, escalabilidade, etc.
   - Exemplos: Testes de carga, testes de estresse, testes de volume de dados.

8. **Testes de Conformidade:**
   - Verificar se o sistema está em conformidade com regulamentações, padrões e requisitos legais.
   - Exemplos: Testes de conformidade com a LGPD, GDPR, PCI-DSS, etc.

9. **Testes de Backup e Recuperação:**
   - Garantir que os dados do sistema possam ser protegidos e recuperados adequadamente em caso de falha.

- Exemplos: Testes de backup, testes de restauração, testes de redundância.

REQUESITOS FUNCIONAIS, NÃO FUNCIONAIS E DO SISTEMA

1. **Requisitos Funcionais:**

- O sistema deve permitir que os usuários cadastrem novos fornecedores, incluindo informações como nome da empresa, CNPJ, endereço completo, contato e informações de pagamento.
- O sistema deve possibilitar o cadastro de novos produtos, incluindo nome, descrição, categoria, preço unitário, quantidade em estoque, data de validade e fornecedor associado.
- Deve ser possível cadastrar novos clientes no sistema, incluindo nome completo, CPF, endereço completo e informações de contato.
- O sistema deve permitir que os usuários registrem novos pedidos, especificando a data do pedido, cliente associado e status do pedido.
- Deve ser possível adicionar itens a um pedido, selecionando o produto, a quantidade e o preço unitário.
- O sistema deve permitir a geração de relatórios de produção, vendas e estoque, com base nos dados registrados no sistema.
- Deve ser possível realizar a autenticação no sistema, utilizando nome de usuário e senha.
- O sistema deve permitir o controle de acesso com base no nível de permissão do usuário (administrador, funcionário).
- Deve ser possível registrar o acesso dos usuários ao sistema, incluindo data e hora do acesso, endereço IP e tipo de acesso (login bem-sucedido, tentativa de login mal-sucedida).

2. **Requisitos Não Funcionais:**

- O sistema deve garantir a segurança dos dados, utilizando criptografia para proteger informações sensíveis.
- Deve ser intuitivo e de fácil utilização, com uma interface amigável e bem projetada.
- O sistema deve ser responsivo, funcionando de forma eficiente em diferentes dispositivos e tamanhos de tela.
- Deve possuir uma alta disponibilidade, garantindo que esteja sempre acessível para os usuários.
- O sistema deve ter um desempenho satisfatório, com tempos de resposta rápidos mesmo sob carga pesada.
- Deve ser escalável, permitindo que seja facilmente expandido para suportar um maior volume de dados e usuários.
- Deve ser compatível com múltiplos navegadores web, garantindo uma experiência consistente para todos os usuários.

3. **Requisitos do Sistema:**

- O sistema deve ser desenvolvido utilizando a linguagem de programação C#.
- A aplicação web deve ser desenvolvida utilizando a tecnologia ASP.NET.
- A aplicação mobile deve ser desenvolvida preferencialmente em Java, com foco em dispositivos Android.
- O banco de dados utilizado deve ser o MS SQL Server, hospedado em um servidor Windows Server.
- O sistema deve ser aderente à LGPD (Lei Geral de Proteção de Dados), garantindo a privacidade e segurança das informações dos usuários.
- Deve ser utilizado controle de versão para o desenvolvimento do sistema, utilizando uma ferramenta como o Git.
- O sistema deve ser desenvolvido seguindo práticas de programação limpa e boas práticas de desenvolvimento de software.

METRICAS DO TRABALHO

1. **Produtividade da Equipe:**
   - Métrica: Número de tarefas concluídas por unidade de tempo (por exemplo, por semana).
   - Como testar: Registre o número de tarefas concluídas pela equipe em um determinado período de tempo e compare com períodos anteriores para verificar a tendência de produtividade.
2. **Qualidade do Código:**
   - Métrica: Taxa de defeitos por linha de código ou por módulo.
   - Como testar: Realize revisões de código regulares e utilize ferramentas de análise estática de código para identificar e corrigir defeitos. Registre o número de defeitos encontrados e corrigidos ao longo do tempo.
3. **Tempo de Resolução de Problemas:**
   - Métrica: Tempo médio necessário para resolver problemas ou bugs relatados.
   - Como testar: Registre o tempo desde o momento em que um problema é identificado até que seja completamente resolvido. Acompanhe esse tempo ao longo do ciclo de vida do projeto e identifique áreas de melhoria.
4. **Satisfação do Cliente:**
   - Métrica: Pesquisas de satisfação do cliente ou feedback direto dos usuários.
   - Como testar: Realize pesquisas regulares de satisfação do cliente ou colete feedback dos usuários sobre a usabilidade, desempenho e funcionalidade do sistema. Analise os resultados e identifique áreas para melhorar a experiência do usuário.
5. **Cobertura de Testes:**
   - Métrica: Porcentagem de código ou funcionalidades cobertas por testes automatizados.
   - Como testar: Utilize ferramentas de cobertura de código para identificar quais partes do código são testadas e quais não são. Estabeleça metas de cobertura de teste e monitore o progresso em direção a essas metas.
6. **Tempo de Resposta do Sistema:**

- Métrica: Tempo médio de resposta do sistema a solicitações do usuário.
- Como testar: Realize testes de desempenho para medir o tempo de resposta do sistema sob diferentes cargas de trabalho. Analise os resultados para identificar gargalos de desempenho e otimizar o sistema conforme necessário.

7. **Disponibilidade do Sistema:**
   - Métrica: Porcentagem de tempo em que o sistema está disponível para uso.
   - Como testar: Utilize ferramentas de monitoramento para registrar o tempo de inatividade do sistema e calcular a disponibilidade com base nesses dados. Estabeleça metas de disponibilidade e monitore o sistema para garantir que essas metas sejam atingidas.

Diagrama de Caso de Uso Geral:

Descrição dos Casos de Uso:

1. **Gerenciar Fornecedores:**
   - Ator Principal: Administrador
   - Breve Descrição: Permite ao administrador adicionar, visualizar, editar e excluir informações de fornecedores, como nome da empresa, CNPJ, endereço, informações de contato e informações de pagamento.
2. **Gerenciar Produtos:**
   - Ator Principal: Administrador
   - Breve Descrição: Permite ao administrador adicionar, visualizar, editar e excluir informações de produtos, como nome, descrição, categoria, preço, quantidade em estoque, data de validade e fornecedor associado.
3. **Gerenciar Clientes:**
   - Ator Principal: Administrador
   - Breve Descrição: Permite ao administrador adicionar, visualizar, editar e excluir informações de clientes, como nome completo, CPF, endereço, informações de contato e histórico de pedidos.
4. **Gerenciar Pedidos:**
   - Ator Principal: Administrador
   - Breve Descrição: Permite ao administrador visualizar detalhes de pedidos, incluindo data do pedido, cliente, status do pedido e itens associados.
5. **Processar Pedidos:**
   - Ator Principal: Funcionário
   - Breve Descrição: Permite ao funcionário processar pedidos, alterando seu status de pendente para em processamento e, em seguida, para concluído após o envio.
6. **Gerar Relatórios:**
   - Ator Principal: Administrador
   - Breve Descrição: Permite ao administrador gerar relatórios sobre diferentes aspectos do sistema, como produção, vendas e estoque, fornecendo uma descrição e salvando o arquivo digitalmente.
7. **Gerenciar Usuários:**
   - Ator Principal: Administrador

- Breve Descrição: Permite ao administrador adicionar, visualizar, editar e excluir informações de usuários, como nome de usuário, senha, nível de acesso, último acesso e status de ativação.

8. **Registrar Acesso do Usuário:**
   - Ator Principal: Sistema
   - Breve Descrição: Registra cada acesso de usuário ao sistema, incluindo o ID do usuário, data e hora do acesso, endereço IP e tipo de acesso (login bem-sucedido ou tentativa de login mal-sucedida).
9. **Configurar Sistema:**
   - Ator Principal: Administrador
   - Breve Descrição: Permite ao administrador configurar parâmetros do sistema, como taxa de impostos e limite de estoque mínimo, definindo valores adequados para esses parâmetros.
10. **Gerenciar Funcionários:**
    - Ator Principal: Administrador
    - Breve Descrição: Permite ao administrador adicionar, visualizar, editar e excluir informações de funcionários, como nome completo, CPF, cargo, data de contratação, salário, departamento e supervisor.

Diagrama de Caso de Uso Geral:

- **Gerenciar Fornecedores:**
  - Ator Principal: Administrador
  - Inclui:
    - Adicionar Fornecedor
    - Visualizar Fornecedor
    - Editar Fornecedor
    - Excluir Fornecedor
  - Generalização:
    - Gerenciar Entidade
- **Gerenciar Produtos:**
  - Ator Principal: Administrador
  - Inclui:
    - Adicionar Produto
    - Visualizar Produto
    - Editar Produto
    - Excluir Produto
  - Generalização:
    - Gerenciar Entidade
- **Gerenciar Clientes:**
  - Ator Principal: Administrador
  - Inclui:
    - Adicionar Cliente

        - Visualizar Cliente
        - Editar Cliente
        - Excluir Cliente
    - Generalização:
        - Gerenciar Entidade
- **Gerenciar Pedidos:**
    - Ator Principal: Administrador
    - Inclui:
        - Visualizar Detalhes do Pedido
    - Generalização:
        - Gerenciar Entidade
- **Processar Pedidos:**
    - Ator Principal: Funcionário
    - Estende:
        - Gerenciar Pedidos
- **Gerar Relatórios:**
    - Ator Principal: Administrador
- **Gerenciar Usuários:**
    - Ator Principal: Administrador
    - Inclui:
        - Adicionar Usuário
        - Visualizar Usuário
        - Editar Usuário
        - Excluir Usuário
    - Generalização:
        - Gerenciar Entidade
- **Registrar Acesso do Usuário:**
    - Ator Principal: Sistema
- **Configurar Sistema:**
    - Ator Principal: Administrador
- **Gerenciar Funcionários:**
    - Ator Principal: Administrador
    - Inclui:
        - Adicionar Funcionário
        - Visualizar Funcionário
        - Editar Funcionário
        - Excluir Funcionário
    - Generalização:
        - Gerenciar Entidade

Descrição dos Relacionamentos:

- Inclui: Representa uma relação de inclusão entre casos de uso, onde um caso de uso inclui outro caso de uso como parte de sua funcionalidade principal. Por exemplo, o caso de uso "Gerenciar Fornecedores" inclui os casos de uso "Adicionar Fornecedor", "Visualizar Fornecedor", "Editar Fornecedor" e "Excluir Fornecedor".
- Estende: Representa uma relação de extensão entre casos de uso, onde um caso de uso estende outro caso de uso adicionando funcionalidades opcionais. Por exemplo, o caso de uso "Processar Pedidos" estende o caso de uso "Gerenciar Pedidos" adicionando a funcionalidade de processamento de pedidos.
- Generalização: Representa uma relação de generalização entre casos de uso, onde um caso de uso genérico é especializado por outros casos de uso mais específicos. Por exemplo, todos os casos de uso de gerenciamento de entidades (fornecedores, produtos, clientes, etc.) são especializados a partir do caso de uso genérico "Gerenciar Entidade".

COISAS DE ENG DE SOFT Q EU PRECISO COLOCAR:

1. **Documentação Adequada:**
   - Além dos diagramas, é essencial fornecer uma documentação detalhada do projeto, incluindo descrições de requisitos, arquitetura do sistema, decisões de design, e assim por diante.
2. **Testes de Software:**
   - Implemente uma estratégia de testes abrangente que cubra diferentes tipos de testes, como testes unitários, testes de integração, testes de sistema e testes de aceitação.
3. **Padrões de Codificação:**
   - Adote padrões de codificação consistentes para garantir a qualidade e a legibilidade do código fonte. Isso pode incluir convenções de nomenclatura, estilo de código e boas práticas de programação.
4. **Gestão de Configuração:**
   - Utilize ferramentas de controle de versão, como Git, e estabeleça um processo claro de controle de mudanças para gerenciar o código fonte e outras artefatos do projeto.
5. **Gerenciamento de Projeto:**
   - Aplique metodologias de gerenciamento de projeto, como Scrum, Kanban ou Waterfall, dependendo das necessidades e características do projeto. Mantenha um planejamento eficaz, atribuição de tarefas e acompanhamento do progresso.
6. **Segurança da Informação:**
   - Implemente medidas de segurança adequadas para proteger o sistema contra ameaças, como autenticação de usuários, autorização de acesso, criptografia de dados e prevenção de ataques cibernéticos.
7. **Usabilidade e Experiência do Usuário:**
   - Realize testes de usabilidade para garantir que o sistema seja intuitivo e fácil de usar. Considere a experiência do usuário em todos os aspectos do design e da interação com o sistema.
8. **Manutenibilidade e Escalabilidade:**

- Projete o sistema com foco na facilidade de manutenção e na capacidade de escalabilidade. Isso inclui a modularização do código, o uso de padrões de projeto e a consideração de requisitos futuros de expansão.

9. **Monitoramento e Diagnóstico:**
   - Implemente ferramentas de monitoramento e diagnóstico para acompanhar o desempenho do sistema, identificar possíveis problemas e tomar medidas corretivas rapidamente.
10. **Feedback e Melhoria Contínua:**
    - Esteja aberto ao feedback dos usuários e das partes interessadas, e use esse feedback para melhorar continuamente o sistema. Mantenha um ciclo de desenvolvimento iterativo e incremental para incorporar melhorias de forma regular.
