**METODOLOGIAS NO DESENVOLVIMENTO**

**Questionário para elicitação de requisitos:**

Sugestões para formular um questionário que ajudará na conversa com os proprietários, funcionários e outras pessoas envolvidas nas operações diárias do negócio:

Informações Gerais:

Nome do Respondente.

Cargo ou Função na Empresa.

Tempo de Experiência na Empresa.

Visão Geral do Negócio:

Como você descreveria o mercadinho em termos de tamanho e escopo?

Quais são os principais produtos ou categorias de produtos que você vende?

Operações Diárias:

Descreva os principais processos diários que ocorrem no mercadinho.

Como é o fluxo de trabalho desde a chegada dos produtos até a venda aos clientes?

Sistemas Atuais (se houver):

Vocês utilizam algum sistema informatizado atualmente? Se sim, como ele é utilizado nas operações diárias?

Quais são os pontos fortes e as limitações do sistema atual?

Não Havendo (Sistemas):

Como vocês gostariam de utilizar algum sistema informatizado? como gostaria de utilizá-lo nas operações diárias?

Gestão de Estoque:

Como é realizado o controle de estoque atualmente?

Quais são os desafios enfrentados na gestão de estoque?

Vendas e Pagamentos:

Como são processadas as vendas?

Como são registrados os pagamentos dos clientes?

Como são registrados os pagamentos dos fornecedores?

Relatórios Financeiros:

Como são gerados os relatórios financeiros atualmente?

Quais são as principais informações que vocês precisam para acompanhar o desempenho financeiro?

Desafios e Obstáculos:

Quais são os principais desafios que o mercadinho enfrenta atualmente?

Existem processos específicos que são mais problemáticos?

-Caso queira.

Feedback dos Clientes:

Como vocês coletam feedback dos clientes?

Existem sugestões ou reclamações recorrentes dos clientes?

Segurança e Privacidade:

Como são tratadas as questões de segurança, especialmente em relação aos dados dos clientes e transações?

Expectativas para um Novo Sistema:

Quais são as suas expectativas em relação a um novo sistema de PDV?

Quais funcionalidades você considera essenciais para facilitar as operações diárias?

\*Treinamento e Adoção:

Como você imagina que os funcionários se adaptarão a um novo sistema?

Quais seriam as considerações importantes para o treinamento da equipe?

Lembre-se de adaptar essas perguntas de acordo com o contexto específico do mercadinho e as informações que você deseja obter. A interação direta com os stakeholders é crucial para entender profundamente as necessidades do negócio e garantir o sucesso do seu projeto.

**PARA MELHOR IDENTIFICAR OS PROCESSOS-CHAVE, VOCÊ PODE SEGUIR AS SEGUINTES ETAPAS:**

Revisão da Entrevista:

Volte para a transcrição ou as notas da entrevista com a proprietária. Identifique quaisquer menções a processos específicos ou atividades que são vitais para o funcionamento diário do mercadinho.

Mapeamento de Fluxo de Trabalho:

Crie um mapa de fluxo de trabalho visual para representar as diferentes etapas envolvidas nas operações diárias. Comece com a chegada dos produtos ao mercadinho e siga até a conclusão da venda e registro do pagamento.

Conversas com Outros Funcionários:

Entreviste outros funcionários para obter diferentes perspectivas sobre os processos. Pode haver diferenças na forma como diferentes equipes ou membros da equipe executam suas tarefas.

Observação Direta:

Se possível, observe diretamente as operações diárias. Isso pode oferecer insights valiosos sobre como os processos são realmente executados, identificando detalhes que podem não ter sido mencionados durante a entrevista.

Identificação de Padrões e Repetições:

Observe padrões e atividades que se repetem com frequência. Isso pode indicar processos-chave que impactam diretamente a eficiência operacional.

Priorização de Atividades Críticas:

Classifique as atividades identificadas de acordo com sua importância para o funcionamento do negócio. Isso ajudará na priorização durante o desenvolvimento do sistema.

Identificação de Pontos de Decisão:

Identifique pontos de decisão críticos no fluxo de trabalho. Estes são momentos em que decisões específicas são tomadas e que podem ter um impacto significativo nas operações.

Análise de Desempenho Atual:

Avalie o desempenho atual de cada processo. Identifique quais processos estão funcionando bem e quais podem ser aprimorados para aumentar a eficiência.

Feedback Iterativo com a Proprietária e Equipe:

Volte a discutir os processos identificados com a proprietária e, se possível, com a equipe. Certifique-se de que sua compreensão está alinhada com as expectativas e que não há detalhes críticos que foram perdidos.

Documentação Detalhada:

Documente de maneira detalhada cada processo identificado. Isso servirá como base para o design do sistema e garantirá que nenhum aspecto importante seja negligenciado.

Ao seguir essas etapas, você terá uma visão mais clara dos processos-chave que precisam ser considerados ao desenvolver o sistema de PDV para o mercadinho. Esse entendimento profundo dos processos garantirá que o sistema atenda efetivamente às necessidades do negócio.

**ESCOLHA DA ARQUITETURA DE SOFTWARE.**

# Dada a ausência de recursos iniciais e considerando que você está começando um projeto para um mercadinho pequeno, uma arquitetura monolítica pode ser uma escolha adequada. Aqui estão alguns motivos para considerar essa abordagem:

Simplicidade Inicial:

Uma arquitetura monolítica é mais simples de desenvolver e implantar inicialmente. Isso pode ser vantajoso quando os recursos são limitados e a prioridade é lançar rapidamente um sistema funcional.

Menor Complexidade:

Uma única aplicação monolítica é mais fácil de entender e manter para desenvolvedores iniciantes. Isso pode ser benéfico, especialmente se você estiver desenvolvendo o sistema sozinho.

Custo Inicial Mais Baixo:

Não há necessidade de infraestrutura complexa ou serviços de gerenciamento de microservices, o que pode reduzir os custos iniciais.

Escalabilidade Futura:

Para um mercadinho pequeno, é provável que a complexidade e escala permaneçam gerenciáveis mesmo com uma arquitetura monolítica. Caso o sistema cresça significativamente, você poderá considerar migrações futuras.

Facilidade de Implantação:

A implantação de uma aplicação monolítica é mais direta, o que é benéfico quando os recursos para gerenciamento de infraestrutura são limitados.

Lembre-se de que, embora a escolha inicial possa ser uma arquitetura monolítica, isso não significa que você estará preso a ela para sempre. À medida que o sistema cresce e os recursos se tornam disponíveis, você pode considerar migrações para arquiteturas mais distribuídas, como microservices, se a situação o exigir.

Independentemente da escolha, é importante manter uma boa organização do código e seguir boas práticas de desenvolvimento para facilitar futuras atualizações e expansões.

A arquitetura monolítica em 3 camadas é um padrão de design que organiza a aplicação em três partes distintas, cada uma com responsabilidades específicas. Essas camadas são:

Camada de Apresentação (ou Interface do Usuário):

Esta camada é responsável por interagir diretamente com o usuário. Ela exibe informações, coleta entrada do usuário e encaminha as solicitações para a camada de lógica de negócios. No contexto de um sistema de PDV para um mercadinho, essa camada seria a interface do usuário que os operadores do caixa interagem para realizar vendas, verificar estoque etc.

Camada de Lógica de Negócios (ou Camada de Aplicação):

Aqui reside a lógica de negócios da aplicação. Ela processa as solicitações provenientes da camada de apresentação, realiza as operações necessárias e interage com a camada de dados para recuperar ou persistir informações. Para um sistema de PDV, essa camada lidaria com operações como registro de vendas, cálculos de estoque e geração de relatórios.

Camada de Dados (ou Armazenamento de Dados):

Esta camada é responsável pelo armazenamento e recuperação de dados. Pode envolver um banco de dados relacional, como MySQL, para armazenar informações sobre produtos, vendas, operadores, etc. Essa camada é essencial para garantir que os dados sejam persistentes e estejam disponíveis quando necessário.

Implementação no Seu Projeto:

Camada de Apresentação:

Desenvolva uma interface do usuário intuitiva e amigável para os operadores do caixa. Pode ser uma aplicação de desktop simples, uma aplicação web leve ou até mesmo uma interface de terminal, dependendo das preferências e requisitos do seu cliente.

Camada de Lógica de Negócios:

Implemente a lógica de negócios que gerencia as operações diárias do mercadinho. Isso incluiria funcionalidades como registro de vendas, gestão de estoque, autenticação de usuários, etc. Certifique-se de manter essa camada coesa e modulada para facilitar a manutenção e expansão.

Camada de Dados:

Escolha um sistema de gerenciamento de banco de dados, como MySQL, para armazenar informações essenciais. Projete o esquema de banco de dados para incluir tabelas para produtos, vendas, operadores etc. Garanta que a interação com o banco de dados seja segura e eficiente.

Comunicação entre Camadas:

Defina interfaces claras e contratos entre as camadas. A camada de apresentação deve chamar os serviços da camada de lógica de negócios, que, por sua vez, interage com a camada de dados para recuperar ou persistir informações.

Segurança:

Implemente mecanismos de segurança, especialmente na camada de apresentação (autenticação) e na camada de lógica de negócios (autorização). Proteger o acesso às funcionalidades críticas é fundamental.

Lembre-se de documentar adequadamente a arquitetura e os processos para facilitar a manutenção e futuras expansões. Uma arquitetura monolítica bem projetada pode ser uma escolha pragmática e eficaz para um projeto inicial com recursos limitados.

**DESENVOLVIMENTO DO SOFTWARE**

Estão os passos iniciais para começar a construir o sistema:

1. Configuração do Ambiente de Desenvolvimento:

Certifique-se de ter um ambiente de desenvolvimento configurado. Isso inclui a instalação das ferramentas necessárias, como o Python e um sistema de gerenciamento de banco de dados (por exemplo, MySQL).

2. Inicialização do Projeto:

Inicie um novo projeto, organize a estrutura de pastas e arquivos. Utilize um sistema de controle de versão, como o Git, para rastrear as alterações no código.

3. Desenvolvimento Incremental:

Adote uma abordagem incremental para o desenvolvimento. Comece implementando funcionalidades essenciais, como fazer login, verificar estoque e cadastrar produtos. Isso permitirá que você tenha uma versão funcional inicial rapidamente.

4. Prototipagem da Interface do Usuário:

Desenvolva protótipos ou interfaces de usuário básicas para as funcionalidades principais. Isso ajudará a visualizar e refinar a experiência do usuário desde o início.

5. Implementação da Lógica de Negócios:

Concentre-se na implementação da lógica de negócios para cada funcionalidade. Isso inclui a gestão de operadoras, gestão de estoque, registro de vendas e geração de relatórios.

6. Integração com Banco de Dados:

Conecte seu sistema à camada de dados, utilizando o MySQL ou o banco de dados escolhido. Implemente operações de CRUD (Create, Read, Update, Delete) para interagir com as informações armazenadas.

7. Implementação de Segurança:

Implemente medidas de segurança, incluindo a autenticação de usuários e a autorização de acesso a diferentes funcionalidades. Considere também a segurança do banco de dados.

8. Testes Unitários e de Integração:

Desenvolva testes unitários para cada componente do sistema e teste a integração entre eles. Isso ajudará a garantir que cada parte funcione conforme o esperado.

9. Iteração e Ajustes:

Esteja preparado para iterar sobre o código à medida que encontra melhorias ou ajustes necessários. A fase de desenvolvimento é iterativa, e você pode aprimorar o sistema à medida que avança.

10. Documentação:

Continue a documentar o código à medida que avança. Isso inclui comentários no código, documentação interna e documentação externa para futuros desenvolvedores ou usuários do sistema.

11. Implantação em Ambiente de Testes:

Implante o sistema em um ambiente de testes para realizar testes mais abrangentes antes de disponibilizá-lo para uso real.

12. Coleta de Feedback:

Colha feedback de usuários ou de pessoas envolvidas no projeto para fazer ajustes finais e refinamentos.

Lembre-se, esse é um guia geral e pode ser ajustado conforme as necessidades específicas do seu projeto. O desenvolvimento é um processo dinâmico, e cada projeto pode ter nuances únicas.

**SOBRE A MODELAGEM DE DADOS**

Após a conclusão do modelo conceitual, o próximo passo é transformá-lo em um modelo lógico. Aqui estão algumas etapas essenciais para criar um modelo lógico bem-feito:

Identificação de Entidades e Atributos:

Refine e amplie a lista de entidades e seus atributos a partir do modelo conceitual. Certifique-se de identificar corretamente as propriedades de cada entidade.

Relacionamentos e Cardinalidade:

Defina os relacionamentos entre as entidades e estabeleça a cardinalidade (um para um, um para muitos, muitos para muitos). Especificações claras dos relacionamentos ajudam na integridade referencial.

Normalização:

Aplique técnicas de normalização para organizar os dados e minimizar redundâncias. Isso envolve dividir tabelas grandes em tabelas menores e mais relacionadas.

Atributos e Tipos de Dados:

Especifique os tipos de dados para cada atributo. Considere as características e requisitos dos dados para escolher o tipo de dado apropriado (inteiro, texto, data, etc.).

Chaves Primárias e Secundárias:

Identifique as chaves primárias para cada tabela, garantindo que sejam únicas e identifiquem de maneira exclusiva cada registro. Considere a necessidade de chaves secundárias conforme necessário.

Índices:

Avalie e identifique quais campos exigem índices para otimizar o desempenho das consultas. Índices aceleram a busca e recuperação de dados.

Restrições de Integridade:

Defina e aplique restrições de integridade referencial para garantir que os dados estejam alinhados com as regras de negócios e evitem inconsistências.

Documentação:

Mantenha uma documentação clara e completa para o modelo lógico, explicando as escolhas feitas em termos de estrutura, relacionamentos, tipos de dados e outras decisões de design.

Revisão e Validação:

Realize revisões regulares do modelo lógico com a equipe de desenvolvimento e stakeholders para garantir que ele atenda aos requisitos do sistema.

Ajustes Iterativos:

Este processo muitas vezes envolve ajustes iterativos. Esteja preparado para revisitar e refinar o modelo lógico à medida que o desenvolvimento progride e novos insights são obtidos.

Lembre-se de que o modelo lógico é uma representação intermediária antes da implementação física do banco de dados, e sua qualidade influenciará diretamente a eficiência e a eficácia do sistema.

**TESTES DO BANCO DE DADOS**

Existem várias etapas de teste que você pode realizar para garantir que seu banco de dados esteja funcionando corretamente e que atenda às suas necessidades. Aqui estão alguns testes que você pode considerar:

1. Teste de Conexão: Verifique se você pode se conectar ao seu banco de dados usando as credenciais corretas. Isso garante que sua configuração de conexão esteja correta.
2. Teste de CRUD: Execute operações de CRUD (Create, Read, Update, Delete) em suas tabelas. Isso envolve inserir dados novos, recuperar dados existentes, atualizar registros e excluir registros. Verifique se as operações são executadas corretamente e se os dados são persistidos conforme esperado.
3. Teste de Restrições: Certifique-se de que as restrições definidas em seu banco de dados estejam funcionando corretamente. Isso inclui restrições de chave primária, restrições de chave estrangeira, restrições de not null, restrições de unicidade, etc. Tente violar essas restrições intencionalmente para garantir que seu banco de dados as aplique corretamente.
4. Teste de Consulta: Execute consultas em suas tabelas para garantir que você possa recuperar os dados necessários de maneira eficiente e precisa. Teste consultas simples e consultas mais complexas que envolvam junções, agregações, ordenações, etc.
5. Teste de Desempenho: Avalie o desempenho do seu banco de dados executando consultas em grandes conjuntos de dados e verificando se as respostas são rápidas o suficiente para atender às suas necessidades. Considere otimizar consultas ou índices, se necessário.
6. Teste de Backup e Recuperação: Faça um backup do seu banco de dados e, em seguida, tente restaurá-lo para garantir que você possa recuperar seus dados em caso de falha.
7. Teste de Segurança: Verifique se as permissões de acesso ao banco de dados estão configuradas corretamente para evitar acesso não autorizado. Teste se diferentes usuários têm acesso apenas às partes do banco de dados que deveriam ter.
8. Teste de Integridade Referencial: Verifique se a integridade referencial entre tabelas está funcionando corretamente. Isso envolve testar a exclusão e atualização em cascata e garantir que os dados estejam sincronizados entre tabelas relacionadas.
9. Teste de Transações: Verifique se transações são tratadas corretamente. Execute operações de transação e verifique se os dados são consistentes após o commit ou rollback.

Esses são apenas alguns dos testes que você pode realizar para garantir a qualidade e integridade do seu banco de dados. A natureza e a extensão dos testes podem variar dependendo dos requisitos específicos do seu projeto e do seu ambiente de banco de dados.

Teste de consulta:

10 exemplos de consultas SQL que você pode usar para testar as seleções envolvendo todas as tabelas, incluindo as tabelas associativas:

1. Selecionar todas as vendas realizadas por uma operadora específica:

sqlCopy code

SELECT \*

FROM tb\_venda

WHERE TB\_Usuaria\_CPF = 'CPF\_da\_operadora';

1. Selecionar todas as vendas de um produto específico:

sqlCopy code

SELECT \*

FROM tb\_venda\_produto

WHERE TB\_Produto\_IdProduto = id\_do\_produto;

1. Selecionar todas as vendas realizadas em uma data específica:

sqlCopy code

SELECT \*

FROM tb\_venda

WHERE DataVenda = '2023-12-25';

1. Selecionar todos os produtos de uma categoria específica:

SELECT \*

FROM tb\_produto

WHERE Categoria = 'categoria\_especifica';

1. Selecionar todas as vendas que envolvem um produto de uma categoria específica:

SELECT v.\*

FROM tb\_venda v

JOIN tb\_venda\_produto vp ON v.idVenda = vp.TB\_Venda\_IdVenda

JOIN tb\_produto p ON vp.TB\_Produto\_IdProduto = p.idProduto

WHERE p.Categoria = 'categoria\_especifica';

1. Selecionar todos os produtos fornecidos por um fornecedor específico:

SELECT \*

FROM tb\_fornecedor\_produto

WHERE TB\_Fornecedor\_IdFornecedor = id\_do\_fornecedor;

1. Selecionar todas as vendas realizadas em uma faixa de valores específica:

sqlCopy code

SELECT \*

FROM tb\_venda

WHERE ValorVenda BETWEEN valor\_minimo AND valor\_maximo;

1. Selecionar todas as vendas realizadas em uma faixa de datas específica:

sqlCopy code

SELECT \*

FROM tb\_venda

WHERE DataVenda BETWEEN 'data\_inicial' AND 'data\_final';

1. Selecionar todas as vendas realizadas com um determinado método de pagamento:

sqlCopy code

SELECT \*

FROM tb\_venda

WHERE TB\_Pagamento\_idPagamento = id\_do\_metodo\_pagamento;

1. Selecionar todas as informações sobre uma venda específica, incluindo detalhes dos produtos vendidos:

sqlCopy code

SELECT v.\*, vp.\*, p.\*

FROM tb\_venda v

JOIN tb\_venda\_produto vp ON v.idVenda = vp.TB\_Venda\_IdVenda

JOIN tb\_produto p ON vp.TB\_Produto\_IdProduto = p.idProduto

WHERE v.idVenda = id\_da\_venda\_especifica;

Essas consultas fornecem uma variedade de cenários que você pode usar para testar e explorar os dados em todas as tabelas do seu banco de dados. Certifique-se de substituir os valores fictícios (como 'CPF\_da\_operadora', 'id\_do\_produto', etc.) pelos valores reais do seu banco de dados

**PLANEJAMENTO DO DESENVOLVIMENTO DA INTERFACE GRÁFICA DO SISTEMA**

Ao planejar o desenvolvimento da interface de usuário em Python usando a biblioteca Tkinter, você pode seguir os seguintes passos:

1. Análise de requisitos: Revise os requisitos do sistema e identifique as funcionalidades que precisam ser suportadas pela interface de usuário. Isso ajudará a determinar quais widgets e elementos da interface precisam ser implementados.
2. Definição de público-alvo: Entenda quem serão os usuários finais da aplicação e suas necessidades específicas. Considere o nível de experiência técnica dos usuários para adaptar a interface de acordo.
3. Levantamento de recursos visuais: Identifique os recursos visuais necessários para a interface, como ícones, imagens e estilos de fonte. Apesar de Tkinter ser mais limitado em termos de personalização visual em comparação com outras bibliotecas, você ainda pode escolher cores e fontes apropriadas para tornar a interface mais atraente.
4. Criação de wireframes ou esboços: Desenvolva wireframes ou esboços básicos que representem a estrutura e o layout da interface de usuário. Isso ajudará a visualizar a disposição dos widgets na tela e a navegação entre diferentes telas ou seções da aplicação.
5. Design da interface: Com base nos wireframes, comece a desenvolver a interface de usuário usando a biblioteca Tkinter. Isso envolve a criação de janelas, frames e widgets (botões, caixas de texto, listas, etc.) conforme necessário para cada tela da aplicação.
6. Implementação da lógica da interface: Desenvolva a lógica por trás da interface de usuário, incluindo a interação com os widgets, validação de entrada do usuário e manipulação de eventos.
7. Testes de usabilidade: Realize testes de usabilidade com usuários reais para avaliar a eficácia, eficiência e satisfação da interface de usuário. Identifique possíveis problemas de usabilidade e faça ajustes no design conforme necessário.
8. Iteração e refinamento: Com base nos feedbacks dos testes de usabilidade e na análise do desempenho da interface, faça iterações e refinamentos no design para melhorar a experiência do usuário e garantir que as necessidades dos usuários sejam atendidas.
9. Integração com o backend: Integre a interface de usuário com o backend do sistema para permitir a funcionalidade completa da aplicação, como a interação com o banco de dados, processamento de dados e comunicação com outros sistemas.

Ao seguir esses passos, você poderá planejar e desenvolver uma interface de usuário eficaz e funcional usando a biblioteca Tkinter em Python. Certifique-se de considerar as necessidades dos usuários finais e testar a usabilidade da interface para garantir uma experiência positiva do usuário.