**Sistema de Pizzaria Online**

**Requisitos Funcionais:**

1. O sistema deve permitir que o cliente faça seu pedido de forma online sem o auxílio de um atendente.
2. O sistema deve permitir que o cliente personalize sua pizza com massa, sabor e borda.
3. O sistema deve incluir um gerenciador de pedidos.
4. O sistema deve armazenar o status do pedido do cliente.
5. O sistema deve ter a ação de remover o pedido do cliente caso seja necessário.
6. O sistema deve ser responsivo.

**Requisitos Não Funcionais:**

1. o sistema deve ser fácil de usar para os clientes e funcionários, com interfaces intuitivas e navegáveis.
2. O sistema deve ser rápido e eficiente, especialmente durante horários de pico de vendas.
3. O sistema deve ser escalável, suportando um grande volume de cadastros.
4. O sistema deve ser capaz de lidar com um grande volume de pedidos e usuários, sem comprometer a qualidade do serviço.

**Product Backlog:**

1. Como cliente, quero que toda a interface seja acessível e intuitiva
2. Como cliente, gostaria de uma mensagem clara de que meu pedido foi realizado com sucesso.
3. Como cliente, gostaria de poder personalizar as minhas pizzas com borda, massa e sabores disponíveis.
4. Como usuário, gostaria de gerenciar os pedidos para entender a demanda da pizzaria.
5. Como usuário, gostaria que o sistema fornecesse informações claras e objetivas sobre as ações disponíveis, para que possa entender como usar o sistema sem precisar de ajuda externa.
6. Como funcionário, gostaria de um dashboard com todas as informações dos pedidos.
7. Como Usuário, quero acompanhar os pedidos em tempo real.
8. Como funcionário, gostaria de remover os pedidos do dashboard caso haja alguma necessidade.

**REQUISITOS ÁGEIS**

**Prof. Artur Marques**

Uma das grandes confusões feitas é quando precisamos criar requisitos ágeis a partir do que convencionamos chamar de Cartões de História do Usuário.

Com o intuito em auxiliar vocês em entender e criar esses cartões para o projeto que devem desenvolver para a disciplina PI, segue uma elucidação mais dirigida e prática.

A partir dela deverão desenvolver o BACKLOG do projeto escolhido para depois priorizar nas sprints que serão desenvolvidas e convertidas em cartões de atividades e tarefas de engenharia de software.

Exemplo:

**CARTÃO DE HISTÓRIA DO USUÁRIO;**

|  |  |
| --- | --- |
| Título: | |
|  | • **Implementar uma função de pesquisa de produtos** |
| Descrição: | |
|  | • Como um usuário do site de comércio eletrônico, eu quero ser capaz de pesquisar por produtos para que eu possa encontrar rapidamente o que estou procurando. |
| Critérios de aceitação: | |
|  | * A função de pesquisa deve ser visível em todas as páginas do site. * A pesquisa deve ser executada em tempo real, sem a necessidade de atualizar a página. * Os resultados da pesquisa devem ser precisos e relevantes para a consulta do usuário. * Os resultados da pesquisa devem ser exibidos em uma lista clara e fácil de navegar. * Os usuários devem ter a opção de filtrar os resultados da pesquisa por categoria, preço e outras opções relevantes. * A função de pesquisa deve ser responsiva e funcionar em todos os dispositivos e tamanhos de tela. * A função de pesquisa deve ser testada e validada antes de ser lançada no site. |
| Valor de negócio: | |
|  | • A implementação de uma função de pesquisa eficaz pode melhorar a experiência do usuário e aumentar as vendas, permitindo que os clientes encontrem facilmente o que estão procurando. |
| Pontos de história: | |
|  | • 5 pontos. |

O cartão de história do usuário é a nossa “ponta do iceberg” do projeto para o desenvolvimento do software, porque temos dezenas, senão centenas dessas cartões dependendo do tamanho e complexidade do projeto de software.

Mas como podemos criar as tarefas a partir de um cartão desses.

Vejamos uma decomposição (tarefas e atividades) utilizando o exemplo que dei logo acima:

Título:

* **Implementar uma função de pesquisa de produtos**

Decomposição em tarefas em ordem alfabética (elas geram as tasks que são executadas pelo time ágil, por exemplo como backlog num quadro Kanban:

* Adição da função de pesquisa em todas as páginas do site.
* Configuração do ambiente de desenvolvimento
* Configuração do back-end da função de pesquisa
* Configuração do front-end da função de pesquisa
* Criação da interface de usuário para a função de pesquisa e integração com a API de pesquisa no back-end.
* Criação da interface de usuário para a função de pesquisa.
* Criação de um esquema para a API de pesquisa que permita buscar produtos por diferentes campos (como nome, descrição, preço, etc.).
* Criação de uma API de pesquisa para permitir que a função de pesquisa interaja com o banco de dados de produtos.
* Definição da estrutura de dados do produto que será armazenada no banco de dados.
* Desenvolvimento de opções adicionais de pesquisa, como filtrar por preço, categoria ou outras opções relevantes.
* Desenvolvimento de testes automatizados para verificar se a API de pesquisa está funcionando corretamente.
* Desenvolvimento de uma lógica de pesquisa eficiente para garantir resultados precisos e relevantes.
* Desenvolvimento do back-end da função de pesquisa
* Desenvolvimento do front-end da função de pesquisa
* Identificação de gargalos de desempenho na função de pesquisa.
* Implementação da funcionalidade de pesquisa em tempo real, sem a necessidade de atualizar a página.
* Implementação da funcionalidade de pesquisa em tempo real, utilizando JavaScript ou outras tecnologias de front-end.
* Implementação da lógica de pesquisa para buscar produtos no banco de dados.
* Implementação de filtros e opções adicionais de pesquisa
* Instalação e configuração do ambiente de desenvolvimento para a equipe trabalhar no projeto.
* Integração da função de pesquisa ao site
* Modelagem de dados do produto
* Otimização da consulta de pesquisa para torná-la mais rápida e eficiente.
* Otimização de desempenho
* Realização de testes de usabilidade com usuários reais para garantir que a função de pesquisa seja fácil de usar e eficaz.
* Teste automatizado da interface de usuário para garantir que a função de pesquisa esteja funcionando corretamente.
* Teste da função de pesquisa em diferentes navegadores e dispositivos para garantir que seja responsivo e funcione em todos os dispositivos e tamanhos de tela.
* Teste de desempenho automatizado para verificar se a função de pesquisa está atendendo aos requisitos de tempo de resposta.
* Teste de validação da função de pesquisa para garantir que ela atenda aos critérios de aceitação especificados na história do usuário.
* Teste e validação das opções de pesquisa para garantir que elas sejam precisas e eficazes.
* Teste manual de integração para verificar se a função de pesquisa está aparecendo e funcionando corretamente em todas as páginas.
* Testes e validação
* Tratamento de casos de erro e exceções (por exemplo, quando não há resultados para uma determinada pesquisa).

Como podemos notar, um cartão aparentemente singelo esconde por traz uma complexidade em tasks decompostas que o produzirão como produto de software.

É isso que deve ser feito para cada cartão no projeto do PI.

Não há caminho fácil, há trabalho e numa empresa esse trabalho é muitas vezes maior que o desse projeto. Portanto aproveite a oportunidade e faça com dedicação e dando seu melhor.

Sabemos que além das Histórias do Usuário há outros artefatos maiores que são percebidos a partir da escrita da história do usuário. São cartões que pela sua complexidade precisam ser decompostos mais e mais até virarem histórias do usuário, ou porque o usuário deixou muito genérico ou porque não percebeu a complexidade envolvida. As vezes isso pode em abordagens ágeis partir do product owner, por engano, pressa ou desconhecimento. Por isso pede-se que o squad tenha minimamente experiência e maturidade para poder atuar colocando suas opiniões, então vamos ver o que são TEMAS e ÉPICOS.

# Metodologia para quebrar um épico em histórias do usuário

Existem diversas metodologias para quebrar um épico em histórias do usuário, mas uma abordagem comum é a técnica de "decomposição progressiva".

Essa técnica envolve a quebra do épico em histórias de usuário menores e mais gerenciáveis, seguindo alguns passos simples:

* Identifique as principais funcionalidades do épico o Faça uma lista das principais funcionalidades ou recursos que o épico deve oferecer.
  + Priorize essas funcionalidades de acordo com o valor que cada uma delas proporciona ao usuário ou ao negócio.
* Quebre as funcionalidades em histórias do usuário o Para cada funcionalidade identificada, pense em um cenário específico em que um usuário poderia se beneficiar daquela funcionalidade.
  + Crie uma história de usuário para descrever esse cenário, utilizando a estrutura "como um (usuário), eu quero (funcionalidade), para que (benefício)".
* Refine as histórias de usuário o Analise cada história de usuário e verifique se ela é clara, concisa e abrange apenas uma funcionalidade específica.
  + Verifique se a história de usuário atende aos critérios de "INVEST": Independente, Negociável, Valiosa, Estimável, Pequena e Testável.
* Priorize as histórias de usuário o Classifique as histórias de usuário em uma ordem de prioridade, com base no valor que cada uma delas proporciona ao usuário ou ao negócio.
  + Leve em consideração a dependência entre as histórias de usuário, para garantir que as mais importantes sejam desenvolvidas primeiro.
* Estime as histórias de usuário
  + Estime o esforço necessário para desenvolver cada história de usuário, utilizando uma técnica como pontos de história ou horas.
  + Use essas estimativas para ajudar a planejar a iteração ou sprint de desenvolvimento.

Lembre-se, é um método, muito utilizado, ao seguir esses passos, você poderá quebrar um épico em histórias de usuário menores e mais gerenciáveis, que poderão ser desenvolvidas em iterações ou sprints menores.

Isso permitirá que você entregue valor ao usuário de forma incremental e iterativa, enquanto mantém um controle mais granular sobre o progresso do projeto.

Para quem não entendeu, um exemplo:

Título do Épico: **Melhorias no processo de vendas online**

*Que está por trás dessa inocente declaração?*

*O que o usuário quis dizer com isso?*

**Bem, percebemos que esse épico gerou ao menos 6 histórias do usuário... mas, vamos começar do princípio.**

**Portanto, vamos decompor:**

* Identifique as principais funcionalidades do épico:
  1. Adição de um carrinho de compras ao site
  2. Implementação de um processo de checkout mais eficiente
  3. Integração com uma ferramenta de pagamento online
  4. Adição de opções de envio e entrega de produtos
* Quebre as funcionalidades em histórias do usuário:
  1. Como um usuário, eu quero poder adicionar produtos ao meu carrinho de compras, para que eu possa comprar mais de um produto de uma só vez.
  2. Como um usuário, eu quero poder visualizar e editar meu carrinho de compras, para que eu possa fazer alterações antes de concluir a compra.
  3. Como um usuário, eu quero poder selecionar um método de envio para minha compra, para que eu possa escolher a opção mais adequada às minhas necessidades.
  4. Como um usuário, eu quero poder selecionar um método de pagamento para minha compra, para que eu possa escolher a opção mais adequada às minhas necessidades.
  5. Como um usuário, eu quero receber um e-mail de confirmação da minha compra, para que eu possa ter um registro dela.
  6. Como um usuário, eu quero poder rastrear o status da minha entrega, para que eu possa saber quando receberei meu produto.
* Refine as histórias de usuário:
  1. Adicionar mais detalhes aos cenários de uso, como o fluxo de interação do usuário com o site, os elementos da interface do usuário e os requisitos de entrada/saída de dados.
  2. Certificar-se de que cada história de usuário aborda apenas uma funcionalidade específica e é clara e concisa.
  3. Verificar se cada história de usuário atende aos critérios INVEST.
* Priorize as histórias de usuário:
  1. Classificar as histórias de usuário em ordem de prioridade com base no valor que cada uma delas proporciona ao usuário ou ao negócio.
  2. Identificar as dependências entre as histórias de usuário para garantir que as mais importantes sejam desenvolvidas primeiro.
* Estime as histórias de usuário:
  1. Estimar o esforço necessário para desenvolver cada história de usuário, usando uma técnica como pontos de história ou horas.
  2. Usar essas estimativas para ajudar a planejar a iteração ou sprint de desenvolvimento.

Claro, além da técnica de decomposição progressiva que mencionei e exemplifiquei anteriormente, existem outras técnicas que podem ser usadas para quebrar um épico em histórias de usuário. Algumas delas são, apenas para conhecimento:

* **Mapa de jornada do usuário:** Nesta técnica, você mapeia o processo de interação do usuário com o sistema ou produto, identificando os principais pontos de interação e as necessidades do usuário em cada um desses pontos. A partir daí, você pode identificar as funcionalidades necessárias para atender a essas necessidades e transformá-las em histórias de usuário.
* **Análise de critérios de aceitação:** Nesta técnica, você define os critérios de aceitação que cada funcionalidade ou requisito do sistema deve atender. A partir daí, você pode quebrar o épico em histórias de usuário que atendam a esses critérios.
* **Cenários de uso:** Nesta técnica, você cria cenários de uso para o sistema ou produto, descrevendo a interação do usuário com o sistema em diferentes situações. A partir daí, você pode identificar as funcionalidades necessárias para suportar esses cenários e transformá-las em histórias de usuário.
* **Épico vertical x épico horizontal:** Nesta técnica, você pode escolher entre quebrar o épico em histórias de usuário que abordem uma funcionalidade específica do sistema (épico vertical) ou que abordem uma camada horizontal do sistema (épico horizontal). Por exemplo, um épico vertical pode ser "Adicionar um carrinho de compras ao site", enquanto um épico horizontal pode ser "Melhorar a performance do site".
* **Modelagem de domínio:** Nesta técnica, você identifica as entidades e relações que compõem o domínio do sistema e as transforma em histórias de usuário. Por exemplo, se o sistema é um sistema de gerenciamento de vendas, você pode identificar as entidades de "cliente", "pedido", "produto", entre outras, e transformá-las em histórias de usuário como "Como um cliente, eu quero poder visualizar meus pedidos anteriores".

Outro tema de debate na comunidade é se m TEMA é menor que um ÉPICO ou se um ÉPICO é menor que um TEMA.

Na verdade, o tamanho de um TEMA em relação a um ÉPICO varia dependendo da metodologia ágil ou da equipe de desenvolvimento envolvida.

Algumas metodologias, como o **SAFe (Scaled Agile Framework)**, **definem que um TEMA é maior que um ÉPICO**, enquanto outras metodologias, como o Scrum, não fazem essa distinção clara. Porém, para nós, **seguiremos o que preconiza SAFe**.

De maneira geral, podemos dizer que um TEMA é um conjunto de histórias de usuário relacionadas a um objetivo comum, que pode ser maior ou menor do que um épico, dependendo do contexto em que está sendo usado.

**O importante é que tanto o TEMA quanto o ÉPICO sejam unidades de planejamento** que ajudem a equipe a organizar o trabalho e a priorizar as funcionalidades do sistema de acordo com seu valor para o usuário.

# TEMA sob a visão SAFe

De acordo com o SAFe (Scaled Agile Framework), TEMA é uma abreviação para "Theme" e é uma **unidade de planejamento de alto nível** **que representa um conjunto de Épicos relacionados a um objetivo estratégico**.

Um TEMA é, portanto, maior do que um Épico e pode inclusive, englobar vários Épicos.

**Diferenciação:**

Diferentemente de um Épico, que é uma unidade de planejamento menor e mais detalhada, um TEMA é uma unidade de planejamento maior e mais estratégica, que ajuda a orientar o trabalho da equipe de desenvolvimento para alcançar um objetivo de negócio específico.

Um Épico, por sua vez, é uma unidade de planejamento intermediária que representa um conjunto de Histórias do Usuário relacionadas a uma funcionalidade específica do sistema.

A História do Usuário é a unidade de trabalho mais detalhada do SAFe e representa um requisito de negócio específico que precisa ser implementado no sistema.

A História do Usuário é uma descrição curta e simples de uma funcionalidade específica do sistema, escrita do ponto de vista do usuário, que ajuda a orientar o trabalho da equipe de desenvolvimento em um nível mais operacional e tático.

Mas a pergunta que paira sob sua mente agora deve ser...

OK, mas como decomponho um TEMA em ÉPICOS e épicos em HISTÓRIA DO USUÁRIO?

Vou te apresentar um exemplo prático e você deve seguir as seguintes etapas:

• Vamos lá, para fazermos um trabalho de qualidade, apresentarei a forma para você decompor um TEMA em ÉPICOS e um épico em HISTÓRIAS DO USUÁRIO, a saber:

1. Identifique as necessidades do usuário relacionadas ao TEMA: Comece identificando as necessidades e desejos dos usuários em relação ao TEMA. O objetivo é entender quais são as funcionalidades que o sistema precisa ter para atender essas necessidades.
2. Crie uma lista de Épicos: Com base nas necessidades dos usuários, crie uma lista de Épicos que representem as principais funcionalidades do sistema relacionadas ao TEMA. Cada Épico deve representar um conjunto de funcionalidades relacionadas.
3. Quebre cada Épico em Histórias do Usuário: Para cada Épico, quebre-o em

Histórias do Usuário que representem funcionalidades específicas do sistema.

Cada História do Usuário deve ser escrita do ponto de vista do usuário e deve representar um valor tangível para o usuário.

Um exemplo:

Suponha que estamos trabalhando em um aplicativo de entrega de comida e nosso **TEMA é "Melhorar a experiência do usuário".**

Podemos criar uma lista de Épicos relacionados ao TEMA, como:

* + Épico 1: Melhorar a experiência de busca e seleção de restaurantes o Épico 2: Melhorar a experiência de seleção de itens do menu o Épico 3: Melhorar a experiência de pagamento

Para cada Épico, podemos quebrá-lo em Histórias do Usuário, como:

* + Épico 1: Melhorar a experiência de busca e seleção de restaurantes.
    - História do Usuário 1.1: Permitir que o usuário filtre restaurantes por tipo de cozinha.
    - História do Usuário 1.2: Permitir que o usuário ordene restaurantes por distância ou avaliação
    - História do Usuário 1.3: Mostrar informações sobre horário de funcionamento, avaliação e distância dos restaurantes.
  + Épico 2: Melhorar a experiência de seleção de itens do menu.
    - História do Usuário 2.1: Permitir que o usuário personalize itens do menu (ex: adicionar/remover ingredientes).
    - História do Usuário 2.2: Mostrar informações nutricionais e alérgenos dos itens do menu.
    - História do Usuário 2.3: Permitir que o usuário adicione itens ao carrinho e edite quantidades.
  + Épico 3: Melhorar a experiência de pagamento.
    - História do Usuário 3.1: Permitir que o usuário salve informações de pagamento para futuras compras.
    - História do Usuário 3.2: Oferecer opções de pagamento como cartão de crédito, débito ou PayPal.
    - História do Usuário 3.3: Permitir que o usuário adicione ou remova cupons de desconto no momento do pagamento.

Ao quebrar o TEMA em Épicos e os Épicos em Histórias do Usuário, estamos criando um plano de desenvolvimento mais detalhado e granular, que nos ajuda a entender melhor as necessidades dos usuários e a priorizar o trabalho da equipe de desenvolvimento, o nosso squad.

Provavelmente você deve continuar refletindo, poxa, mas isso é fácil né!?

Como eu quebro esse monte de histórias em tarefas e atividades de execução de projeto?

Pois é, por isso vou pegar uma delas e te dar mais um exemplo, apesar de lá em cima eu já ter feito uma decomposição, aqui vai mais uma.

Vamos ver como é isso:

**A História do Usuário 3.2 é "Oferecer opções de pagamento como cartão de crédito, débito ou PayPal".**

Para quebrá-la em tarefas de engenharia de software e de desenvolvimento, podemos seguir as seguintes etapas:

1. Identifique os requisitos técnicos: Comece identificando os requisitos técnicos necessários para oferecer opções de pagamento como cartão de crédito, débito ou PayPal. Isso pode incluir integrações com serviços de pagamento, segurança de transações e tratamento de erros de pagamento.
2. Quebre em tarefas de engenharia de software: Com base nos requisitos técnicos, quebre a História do Usuário em tarefas de engenharia de software específicas. Algumas tarefas possíveis são: o Tarefa 1: Pesquisar e escolher uma biblioteca de integração com serviços de pagamento, como Stripe ou PayPal.
   * Tarefa 2: Implementar a integração com a biblioteca escolhida, incluindo autenticação e segurança de transações.
   * Tarefa 3: Tratar possíveis erros de pagamento, como cartão inválido ou transação negada.
   * Tarefa 4: Testar e validar a integração de pagamento em diferentes cenários de uso.
3. Quebre em tarefas de desenvolvimento: Com base nas tarefas de engenharia de software, quebre a História do Usuário em tarefas de desenvolvimento mais específicas.

Algumas tarefas possíveis são: o Tarefa 1: Configurar o ambiente de desenvolvimento com a biblioteca de integração escolhida e criar um ambiente de teste para integração de pagamento.

* + Tarefa 2: Implementar a funcionalidade de seleção de opções de pagamento no frontend, incluindo validação de formulário e exibição de mensagens de erro. o Tarefa 3: Implementar a lógica de integração com a biblioteca de pagamento no backend, incluindo tratamento de erros e geração de tokens de segurança.
  + Tarefa 4: Testar e validar a funcionalidade de pagamento em diferentes navegadores e dispositivos, além de validar a integração com o ambiente de teste.

Com essa abordagem de quebrar a História do Usuário em tarefas de engenharia de software e desenvolvimento, podemos ter uma visão mais clara do trabalho necessário para implementar a funcionalidade de seleção de opções de pagamento.

Essa abordagem também pode ajudar a equipe a priorizar as tarefas de acordo com sua complexidade e dependências técnicas.

Por fim, talvez você queira saber como essas tarefas viram cards no quadro de KanBan, certo, certo, vamos lá então.

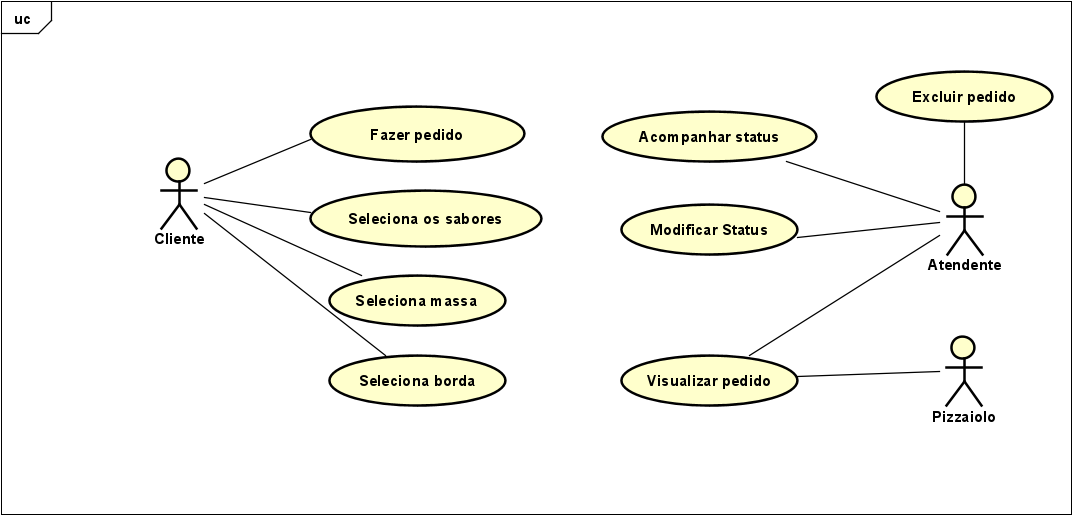
* + Tarefa 1: Pesquisar e escolher uma biblioteca de integração com serviços de pagamento, como Stripe ou PayPal.
  + Cartão Kanban: "Pesquisar bibliotecas de integração de pagamento"
    - Descrição: Pesquisar diferentes bibliotecas de integração de pagamento e avaliar suas vantagens e desvantagens.
  + Tarefa 2: Implementar a integração com a biblioteca escolhida, incluindo autenticação e segurança de transações.
  + Cartão Kanban: "Implementar integração com serviço de pagamento"
    - Descrição: Configurar e integrar a biblioteca escolhida com o sistema, incluindo autenticação e segurança de transações.
  + Tarefa 3: Tratar possíveis erros de pagamento, como cartão inválido ou transação negada.
  + Cartão Kanban: "Tratar erros de pagamento"
    - Descrição: Identificar e tratar possíveis erros de pagamento que possam ocorrer durante o processo de compra.
  + Tarefa 4: Testar e validar a integração de pagamento em diferentes cenários de uso.
  + Cartão Kanban: "Testar integração de pagamento"
    - Descrição: Realizar testes e validações da integração de pagamento em diferentes cenários de uso, garantindo a funcionalidade correta da solução implementada.

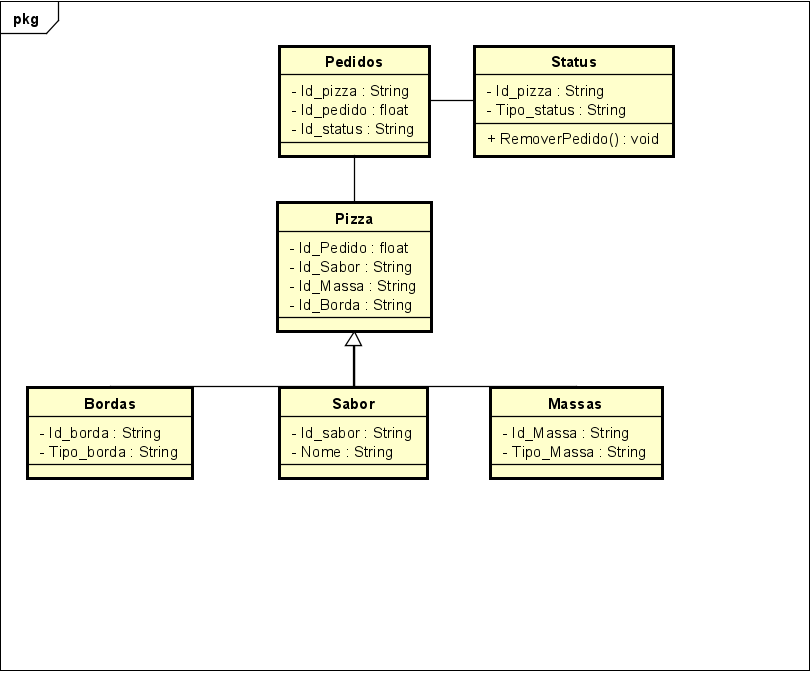
Esses cartões do quadro Kanban podem ser arrastados pelos membros da equipe para diferentes colunas do quadro, como: "Para fazer", "Fazendo", "Testando" e "Concluído", permitindo que a equipe visualize o status de cada tarefa, respeitando o número WIP – *Work in Progress* de cada uma das colunas e priorize o trabalho conforme necessário.

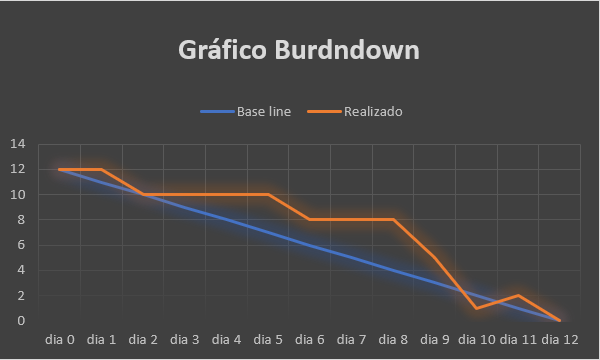
Além disso, é possível incluir outros detalhes relevantes em cada cartão, como dependências, prazos e responsáveis pela tarefa, claro que, conforme o grau de experiência e maturidade de cada equipe e de cada empresa.

Bom, espero ter ajuda.

Aguardo essa confecção completa do projeto escolhido para o PI a ser apresentado aos professores Paulo e Edidio já na próxima aula.







**Sprint Backlog**

**Sprint 1:**

* Criar tela de cadastro de clientes com validação de campos obrigatórios.
* Implementar a funcionalidade de criação de pedidos com opções de personalização de pizza.
* Criar tela de exibição de pedidos em andamento com informações de status.
* Implementar a funcionalidade de remoção de pedidos por parte do cliente.

**Sprint 2:**

* Implementar um gerenciador de pedidos para a equipe de funcionários da pizzaria.
* Adicionar funcionalidade de atualização do status do pedido em tempo real.
* Criar tela de exibição de histórico de pedidos para clientes.
* Implementar a funcionalidade de cancelamento de pedidos por parte dos funcionários.

**Sprint 3:**

* Otimizar o sistema para que seja responsivo em diferentes dispositivos e tamanhos de tela.
* Refatorar o código para melhorar a eficiência e a velocidade do sistema.
* Implementar testes de carga para garantir escalabilidade do sistema.
* Adicionar opções de gerenciamento de usuários para administradores do sistema.

**Introdução:**

* O objetivo desta reunião é revisar a documentação do sprint anterior e discutir a documentação a ser produzida para o próximo sprint.

**Revisão do Sprint:**

* O documento de requisitos foi revisado e aprovado.
* As histórias de usuário e seus critérios de aceitação foram documentados e aprovado.
* O sprint backlog foi criado com base nos requisitos e histórias de usuário.

**Documentação do Sprint:**

* O objetivo deste sprint é desenvolver a documentação necessária para as funcionalidades relacionadas ao gerenciamento de pedidos.
* A documentação deve incluir instruções claras e detalhadas para os clientes e funcionários sobre como usar as funcionalidades do sistema.

Os seguintes documentos devem ser produzidos:

* Manual do Usuário para o cliente: deve incluir as instruções para fazer um pedido online, personalizar uma pizza, gerenciar seu próprio pedido, e verificar o histórico de pedidos.
* Manual do Funcionário para a equipe da pizzaria: deve incluir as instruções para gerenciar os pedidos, atualizar o status do pedido e cancelar um pedido se necessário.
* Documentação técnica: deve incluir descrições técnicas das funcionalidades do sistema, incluindo arquitetura, banco de dados, integrações de sistemas, e outros detalhes técnicos.

**Responsabilidades:**

* O time de desenvolvimento será responsável por criar a documentação técnica e revisar os manuais do usuário e funcionário.
* O time de produto será responsável por revisar e aprovar a documentação produzida.

**Encerramento:**

Story Cards

* A próxima reunião de sprint será agendada após a revisão da documentação.
* Como um cliente, eu gostaria de criar uma conta para que eu possa fazer pedidos e ver meu histórico de pedidos anteriores.
* Como um cliente, eu gostaria de poder fazer login na minha conta para facilitar o processo de checkout.
* Como um cliente, eu gostaria de visualizar o menu completo da pizzaria para que eu possa fazer um pedido completo.
* Como um cliente, eu gostaria de poder filtrar o menu por tipo de pizza, tamanho ou ingredientes para facilitar a busca de opções específicas.
* Como um cliente, eu gostaria de poder personalizar a minha pizza escolhendo a massa, sabor e borda, bem como adicionar ou remover ingredientes.
* Como um cliente, eu gostaria de poder ver a minha pizza personalizada em uma imagem para visualizar como ficará antes de finalizar o pedido.
* Como um cliente, eu gostaria de poder adicionar várias pizzas ao meu carrinho de compras de uma só vez.
* Como um cliente, eu gostaria de poder visualizar o meu carrinho de compras e fazer alterações antes de finalizar o pedido.
* Como um cliente, eu gostaria de poder finalizar o meu pedido e pagar online, utilizando um cartão de crédito ou outra forma de pagamento eletrônico.
* Como um cliente, eu gostaria de poder escolher a forma de entrega do meu pedido, seja por delivery ou retirada na loja.
* Como um funcionário, eu gostaria de receber notificações em tempo real de novos pedidos, para que eu possa começar a preparar a pizza o mais rápido possível.
* Como um funcionário, eu gostaria de poder atualizar o status do pedido, para que o cliente possa acompanhar o andamento do seu pedido em tempo real.
* Como um funcionário, eu gostaria de poder gerenciar o estoque de ingredientes para garantir que tenhamos sempre os ingredientes necessários para as pizzas.
* Como um funcionário, eu gostaria de poder gerenciar as informações dos clientes e seus pedidos anteriores.
* Como um administrador, eu gostaria de poder adicionar ou remover funcionários para gerenciar o sistema.
* Como um administrador, eu gostaria de poder visualizar relatórios de vendas e estatísticas de desempenho para ajudar a tomar decisões de negócios.
* Como um administrador, eu gostaria de poder gerenciar as informações da pizzaria, como endereço, horário de funcionamento e informações de contato.
* Como um administrador, eu gostaria de poder visualizar os logs do sistema para solucionar problemas ou detectar falhas.
* Como um cliente, eu gostaria de poder cancelar o meu pedido caso haja algum problema ou atraso na entrega.
* Como um cliente, eu gostaria de poder avaliar a minha experiência de compra e deixar um feedback sobre a pizzaria.

**Requisitos de negócios:**

* Os clientes devem ser capazes de criar pedidos personalizados de pizza com opções de borda, sabor, cobertura.
* Os clientes devem ser capazes de fazer pedidos online e pagar usando cartão de crédito ou outras opções de pagamento seguras.
* Os funcionários devem ser capazes de gerenciar o estoque de ingredientes e monitorar pedidos.
* O sistema será projetado para ser facilmente extensível para acomodar novos recursos ou integrações futuras.

**Funcionalidade:**

* Os clientes poderão criar pedidos personalizados de pizza e visualizar um resumo do pedido antes de finalizar o pagamento.
* Os clientes poderão visualizar o status do pedido.
* Os clientes poderão visualizar e gerenciar seu histórico de pedidos.

**Usabilidade:**

* A interface do usuário do sistema será intuitiva e fácil de usar, com uma navegação clara e botões e opções claramente rotulados.
* O sistema será responsivo e compatível com dispositivos móveis para permitir o uso em smartphones e tablets.
* O sistema passará por testes de unidade, integração e sistema para garantir que todos os recursos funcionem corretamente e sem erros.

**Desempenho:**

* O sistema será capaz de lidar com cargas de trabalho esperadas sem desempenho lento ou interrupções.
* O sistema será projetado para ter tempos de resposta rápidos e uma experiência do usuário suave.

**Segurança:**

* O sistema será protegido contra ameaças de segurança.
* O sistema terá controles de acesso adequados para garantir que apenas usuários autorizados possam acessar informações confidenciais.

**Documentação:**

* A documentação do sistema será completa, incluindo manuais do usuário, guias de administração e documentação técnica.
* O sistema terá um processo de suporte definido.