Documentação do Código de Automação de Pedido de Pizza

Este código automatiza o processo de coleta de informações de um pedido de pizza, validação do CEP, e envio do pedido para uma fila de trabalho e para outro fluxo.

1. Coleta de Informações do Cliente

- **Objetivo:** Coletar o nome e telefone do cliente.
- **Ações:**
    - `Display.InputDialog`: Exibe uma caixa de diálogo para inserir o nome do cliente.
        - `Title`: "Pedido"
        - `Message`: "Nome:"
        - `InputType`: `Display.InputType.SingleLine` (linha única de texto)
        - `IsTopMost`: `False` (não fica sempre no topo)
        - `UserInput=> Nome`: Armazena o nome inserido na variável `Nome`.
        - `ButtonPressed=> ButtonPressed3`: Armazena o botão pressionado na variável `ButtonPressed3`.
    - `Display.InputDialog`: Exibe uma caixa de diálogo para inserir o telefone do cliente.
        - `Title`: "Pedido"
        - `Message`: "Telefone"
        - `InputType`: `Display.InputType.SingleLine`
        - `IsTopMost`: `False`
        - `UserInput=> Telefone`: Armazena o telefone inserido na variável `Telefone`.
        - `ButtonPressed=> ButtonPressed3`: Armazena o botão pressionado na variável `ButtonPressed3`.
    - `Display.SelectFromListDialog.SelectItemsFromList`: Exibe uma caixa de diálogo para selecionar o sabor da pizza.
        - `Title`: " Sabor"
        - `List`: "Portuguesa\nMussarela\nCalabresa\nQuatro Queijos" (lista de sabores)
        - `IsTopMost`: `False`
        - `AllowEmpty`: `False` (não permite seleção vazia)
        - `PreSelectItemsWithPlus`: `False`
        - `SelectedItems=> Sabor`: Armazena o sabor selecionado na variável `Sabor`.
        - `SelectedIndexes=> Pedido`: Armazena o indice do sabor selecionado na variavel pedido.

        - `ButtonPressed=> ButtonPressed`: Armazena o botão pressionado na variável `ButtonPressed`.

2. Exibição de Resumo do Pedido em Cartão Adaptativo

- **Objetivo:** Exibir um resumo do pedido em um cartão adaptativo para confirmação.
- **Ações:**
    - `Display.ShowCustomDialog`: Exibe um cartão adaptativo personalizado.
        - `CardTemplateJson`: Define a estrutura do cartão adaptativo em formato JSON.
            - Inclui campos de entrada para nome, telefone e sabor, preenchidos com as variáveis coletadas anteriormente.
            - Inclui botões "Confirmar Pedido" e "Cancelar".
            - `CustomFormData=> Pedido`: Armazena os dados inseridos no cartão adaptativo na variável `Pedido`.
            - `ButtonPressed=> ButtonPressed3`: Armazena o botão pressionado na variável `ButtonPressed3`.
            - `@AdaptiveCard_Url`: Define a URL da imagem de fundo do cartão adaptativo.
            - `@Text_input_Label`: Define o label do campo nome.
            - `@Text_input_Value`: Define o valor do campo nome.
            - `@Text_input2_Label`: Define o label do campo telefone.
            - `@Text_input2_Value`: Define o valor do campo telefone.
            - `@Text_input4_Label`: Define o label do campo sabor.
            - `@Text_input4_Value`: Define o valor do campo sabor.
            - `@Submit_Title`: Define o título do botão de confirmar pedido.
    - `@Submit2_Title`: Define o título do botão cancelar.

3. Validação do CEP

- **Objetivo:** Coletar e validar o CEP do cliente usando a API ViaCEP.
- **Ações:**
    - `LABEL 'Preencher CEP'`: Define um rótulo para permitir o retorno a este ponto em caso de CEP inválido.
    - `Display.InputDialog`: Exibe uma caixa de diálogo para inserir o CEP.
        - `Message`: "Digite seu CEP:"
        - `DefaultValue`: `CEP` (usa o valor anterior de `CEP`, se houver)
        - `InputType`: `Display.InputType.SingleLine`

        - IsTopMost: False
        - UserInput=> CEP: Armazena o CEP inserido na variável CEP.
        - ButtonPressed=> ButtonPressed2: Armazena o botão pressionado na variável ButtonPressed2.
    - Web.InvokeWebService.InvokeWebService: Chama a API ViaCEP para obter informações do endereço.
        - Url: "viacep.com.br/ws/%CEP%/json/" (substitui %CEP% pelo valor da variável CEP)
        - Method: Web.Method.Get (requisição GET)
        - Accept: "application/json"
        - ContentType: "application/json"
        - Response=> WebServiceResponse2: Armazena a resposta da API na variável WebServiceResponse2.
        - StatusCode=> StatusCode2: Armazena o código de status da resposta na variável StatusCode2.
    - IF NotStartsWith(StatusCode2, 20, False): Verifica se o código de status não começa com "20" (indicando erro).
        - Display.ShowMessageDialog.ShowMessage: Exibe uma mensagem de erro se o CEP for inválido.
        - GOTO 'Preencher CEP': Retorna ao rótulo "Preencher CEP" para que o cliente insira o CEP novamente.
    - Variables.ConvertJsonToCustomObject: Converte a resposta JSON da API em um objeto personalizado na variável WebServiceResponse2.

4. Envio do Pedido para Fila de Trabalho e outro Fluxo.

- **Objetivo:** Enviar os dados do pedido para uma fila de trabalho e disparar outro flow passando os dados do pedido.
- **Ações:**
    - WorkQueues.EnqueueWorkQueueItem.WithoutUniqueId: Adiciona os dados do pedido a uma fila de trabalho.
        - WorkQueue: Identificador da fila de trabalho.
        - Status: WorkQueues.WorkQueueItemEnqueueStatus.Queued (define o status como "Enfileirado").
        - Priority: WorkQueues.WorkQueueItemPriority.Normal (prioridade normal).
        - Name: "Pedido" (nome do item na fila).
        - Value: JSON contendo nome, telefone, sabor, CEP e endereço do cliente.
        - WorkQueueItem=> Pedido: Armazena o item da fila na variável Pedido.

- `External.RunFlow`: Executa outro fluxo do power automate desktop.
- `FlowId`: Identificador do fluxo a ser executado.
- `WaitToComplete`: `True` (espera a conclusão do fluxo externo).
- `@Nome: Nome`: Passa o nome do cliente para o fluxo externo.
- `@Telefone: Telefone`: Passa o telefone do cliente para o fluxo externo.
- `@Sabor: Sabor`: Passa o sabor da pizza para o fluxo externo.
- `@CEP: CEP`: Passa o CEP do cliente para o fluxo externo.

Observações:

- O código utiliza variáveis para armazenar as informações coletadas e processadas.
- A validação do CEP é realizada através de uma chamada à API ViaCEP.
- O uso de fila de trabalho, permite que o restante do processo de preparo da pizza seja feito de maneira assíncrona, sem a necessidade que o cliente espere.
- O uso de outro fluxo, permite que outras ações sejam executadas, como por exemplo, salvar os dados do pedido em um arquivo excel.
- O trecho de código `DISABLE Web.InvokeWebService.InvokeWebServicePost` indica que existe uma chamada para uma API externa que está desabilitada.

Documentação do Código Processar Fila Pizzaria

Este código processa itens de uma fila de trabalho (Work Queue), exibindo um diálogo personalizado para o usuário escolher o status do pedido e, em seguida, atualizando o item da fila de trabalho com base na escolha do usuário.

Função:

O código itera sobre os itens da fila de trabalho "FilaPedidos" (ID: f0bdd9cf-4ee3-ef11-9341-000d3ac0530f), apresentando um diálogo ao usuário para selecionar o status do pedido (Entregue ou Não Entregue). Dependendo da escolha, o item da fila é atualizado.

Passos:

1. **`@@'InputSummaryValue:WORKQUEUE': 'FilaPedidos'`**: Esta linha é um comentário e indica que a fila de trabalho de entrada é "FilaPedidos".

2. **`LOOP WHILE (WorkQueues.ProcessWorkQueueItem.ProcessWorkQueueItem ...)`**: Este loop itera sobre cada item na fila de trabalho especificada.
    a. `WorkQueue: $'''f0bdd9cf-4ee3-ef11-9341-000d3ac0530f'''`: Especifica o ID da fila de trabalho a ser processada.
    b. `WorkQueueItem=> WorkQueueItem`: Armazena o item atual da fila de trabalho na variável `WorkQueueItem`.
    c. O loop continua enquanto houver itens na fila de trabalho.
3. **`Variables.ConvertJsonToCustomObject ...`**: Converte o valor do item da fila de trabalho em um objeto personalizado (`JsonAsCustomObject`).
    a. `Json: WorkQueueItem.Value`: O valor JSON do item da fila de trabalho.
    b. `CustomObject=> JsonAsCustomObject`: A variável para armazenar o objeto personalizado resultante.
4. **`@@statistics_Input_ChoiceSet: '1'` e `@@statistics_Action_Submit: '2'`**: Estas linhas são comentários e indicam informações para rastreamento de estatísticas sobre as interações do usuário com o diálogo.
5. **`Display.ShowCustomDialog ...`**: Exibe um diálogo personalizado (Adaptive Card) para o usuário.
    a. `CardTemplateJson`: Define a estrutura do diálogo em formato JSON.
        i. `Input.ChoiceSet`: Um campo de seleção única com as opções "Entregue" e "Não Entregue".
        ii. `Action.Submit`: Dois botões de ação: "Ok" e "Cancelar".
    b. `CustomFormData=> CustomFormData`: Armazena os dados do formulário (a escolha do usuário) na variável `CustomFormData`.
    c. `ButtonPressed=> ButtonPressed`: Armazena o botão pressionado pelo usuário na variável `ButtonPressed`.
    d. `@Status_Pedido_Label`, `@Submit_Title`, `@Submit2_Title`: Definem os textos exibidos no diálogo.
6. **`IF JsonAsCustomObject = $'''Entrega''' THEN ... END`**: Verifica se o valor convertido do JSON (`JsonAsCustomObject`) é igual a "Entrega". **`WorkQueues.UpdateWorkQueueItem.UpdateWithProcessingNotes ...`**: Atualiza o item da fila de trabalho com anotações de processamento.
    a. `WorkQueueItem: WorkQueueItem`: O item da fila de trabalho a ser atualizado.
    b. `Status: WorkQueues.WorkQueueItemStatus.BusinessException`: Define o status do item como "Exceção de Negócio".
    c. `ProcessingResult: $'''Sucesso!'''`: Adiciona uma anotação de processamento "Sucesso!".

Em resumo, este código demonstra um fluxo de trabalho para processar itens de uma fila de trabalho, permitindo que um usuário interaja com o sistema através de um diálogo personalizado. O usuário fornece informações que são então usadas para atualizar o item da fila de trabalho, refletindo o status ou resultado do processamento. Este processo automatiza a interação com a fila de trabalho, simplificando e agilizando tarefas que poderiam ser manuais. A utilização de um diálogo personalizado oferece uma interface amigável para o usuário, enquanto a atualização do item da fila de trabalho garante o rastreamento e o registro do progresso.