Nome: Elias Cyrino de Assis
Descrição: detalhamento da solução desenvolvida para o desafio “**processo de pedido de pizza online**”.

Neste documento serão descritas as decisões para escrita da *blueprint* (BP) referentes à solução desenvolvida para o processo solicitado como desafio. Uma breve consideração sobre o processo de pedido de pizza online também será apresentada. Durante a descrição da solução, serão abordados os pontos principais de escrita da BP em relação ao diagrama BPMN proposto, correspondendo cada nó a sua respectiva representação JSON na BP.

Processo de pedido de pizza online

Na solução proposta, o processo de pedido de pizza online se baseou no processo padrão de compra e entrega por aplicativo, bem comum em aplicativos como *ifood* e *ubereats*. No geral, esse processo segue as seguintes etapas:

- Cliente solicita pedido por aplicativo mediante pagamento
- Sistema verifica o pagamento do pedido e o libera para preparação
- Restaurante prepara o pedido e o libera para entrega
- Entregador entrega o pedido para o cliente
- Cliente classifica o serviço de acordo com sua experiência

O diagrama BPMN inspirado no processo acima, encontra-se na Figura 1:

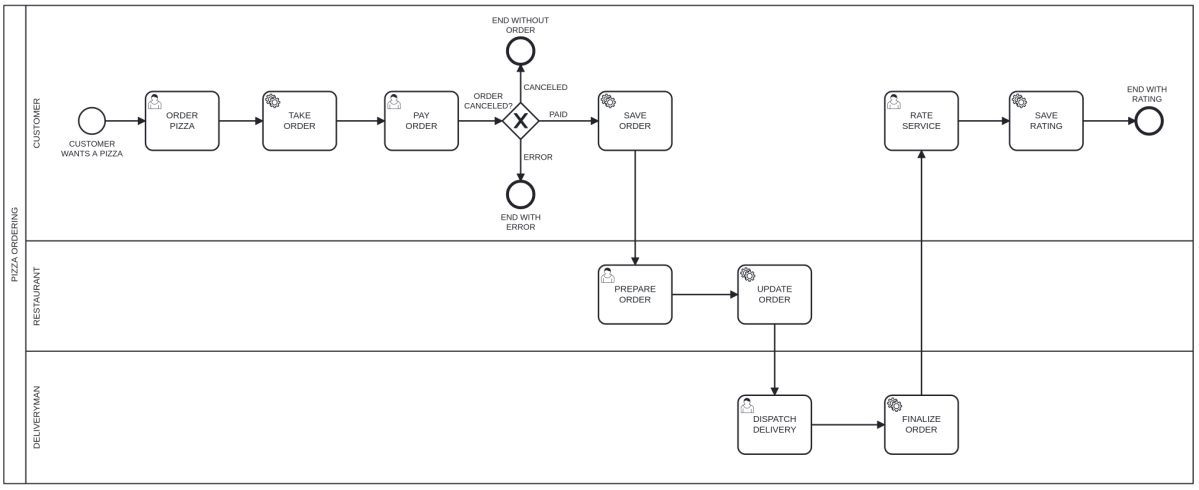

Figura 1: diagrama BPMN do processo de pedido de pizza online

Como mostrado na Figura 1, o processo se inicia com o desejo do cliente de pedir por uma pizza (CUSTOMER WANTS A PIZZA). O cliente realiza o pedido (ORDER PIZZA) e o sistema armazena esses dados no processo (TAKE ORDER). Após essas etapas, o pagamento do pedido é solicitado ao cliente (PAY ORDER). Nesse ponto podem ocorrer três situações (ORDER CANCELED?): o pedido ser cancelado (CANCELED), ocorrer um erro no pagamento do pedido (ERROR) ou o pagamento do pedido ser efetuado com sucesso (PAID). Nos casos onde o pagamento não ocorre o processo se encerra no respectivo nó final (END WITHOUT ORDER ou ERROR). Caso o pagamento do pedido

ocorra com sucesso, o processo segue para a próxima tarefa, onde o pedido é persistido no sistema (SAVE ORDER).

Com o pedido do sistema devidamente armazenado, o pedido é preparado pelo restaurante (PREPARE ORDER) e após a preparação o status do pedido é atualizado no sistema (UPDATE ORDER), informando que o pedido já pode ser entregue. O pedido é então enviado ao entregador para entrega (DISPATCH DELIVERY). Após a entrega, o pedido é finalizado no sistema (FINALIZE ORDER) e o cliente pode então classificar o serviço com base em sua experiência (RATE SERVICE). O sistema persiste os dados da classificação do cliente (SAVE RATING) e o processo se encerra (END WITH RATING).

Considerações sobre o processo proposto como solução

Uma solução mais robusta quanto ao fluxo de pagamento do pedido que considerasse diversas opções de pagamento (PIX, débito/crédito do cartão, boleto, ….) talvez fosse necessária. No entanto, a depender da interpretação, esse fluxo pode ser entendido como um processo à parte, que seria iniciado, por exemplo, devido a um erro de pagamento. A forma como se paga pedidos por aplicativo pode ser entendida como um processo distinto do pagamento do pedido em si. Na solução apresentada, optou-se por considerar que o pagamento é realizado diretamente dos dados já cadastrados pelo cliente no aplicativo.

Outro ponto se dá quanto à entrega do pedido, onde um nó de sistema da categoria *timer* poderia ser colocado na *lane* do entregador (*deliveryman*) entre a entrega do pedido e a finalização do pedido no sistema. Poderia se considerar que, se o pedido demorasse um determinado tempo para ser entregue, que o pagamento poderia ser suspenso e o dinheiro do cliente ressarcido, uma vez que o pedido é pago de antemão. No entanto, devido à complexidade de um processo que considera a devolução do pagamento, optou-se por não incluir essas etapas. Ainda, a solução inicialmente pensada considerava que o cliente poderia executar a ação de cancelamento do pedido a qualquer momento, o que adicionaria certo paralelismo ao processo, motivo pelo qual também foi retirado da solução um acompanhamento do status do pedido por parte do cliente.

A inserção de um nó de sistema da categoria *timer* também impactaria no que foi pensado inicialmente para o cenário da solução, cujo restaurante prepara o pedido assim que possível e entrega o pedido assim que possível. Nesse sentido, de modo semelhante ao que foi feito para o fluxo do pagamento, o cancelamento do pedido e ressarcimento do pagamento poderia ser tratado como um processo distinto, a ser iniciado a qualquer momento pelo cliente. Optou-se então por fechar o escopo da solução somente nas etapas diretamente relacionadas com um fluxo contínuo de pedido de pizza, sem a adoção de quaisquer nós que ocasionassem algum tipo de paralelismo no processo.

Lanes e regras

Como mostrado na Figura 1, a solução proposta é composta por três *lanes*: uma *lane* referente ao cliente (*customer*) do restaurante, uma *lane* referente ao restaurante em si (*restaurant*) e uma *lane* referente ao entregador do pedido de pizza (*deliveryman*). As *lanes* foram configuradas de acordo com o presente na documentação do *flowbuild*. Para a *lane* do cliente, as tarefas do processo foram restringidas a um usuário específico, como mostrado no Trecho de Código 1:

```
{
    "id": "customer",
    "name": "Only the customer can access.",
    "rule": [
         "fn", ["actor_data", "bag"],
         ["=", ["get", ["get", "bag", ["`", "customer"]], ["`", "id"]],
         ["get", "actor_data", ["`", "customer_id"]]]
    ]
}
```

Trecho de Código 1: trecho referente à *lane* do cliente

Já as *lanes* referentes às atribuições do restaurante e do entregador foram configuradas com a mesma regra, no entanto, restringindo o acesso para os usuários cujo o *actor_data* contém a respectiva credencial que permite o acesso na *lane*: *restaurant* para empregados do restaurante e *deliveryman* para entregadores. A configuração dessas *lanes* pode ser vista no Trecho de Código 2:

```
{
    "id": "restaurant",
    "name": "Only restaurant employees can access",
    "rule": [
                 "fn", ["actor_data"],
                 ["eval", ["apply", "or", ["map", ["fn", ["v"],
                 ["=", "v", ["`", "restaurant"]]],
                 ["get", "actor_data", ["`", "claims"]]]]]
    ]
},
{
    "id": "deliveryman",
    "name": "Only deliverymans can access.",
    "rule": [
                "fn", ["actor_data"],
                ["eval", ["apply", "or", ["map", ["fn", ["v"],
                ["=", "v", ["`", "deliveryman"]]],
                ["get", "actor_data", ["`", "claims"]]]]]
    ]
}
```

Trecho de Código 2: trecho referente às *lanes* do restaurante e do entregador

Nós do processo

Nó de início do processo (START-PROCESS)

O primeiro nó apresentado na Figura 1 diz respeito ao evento de início (*startNode*) do processo (CUSTOMER WANTS A PIZZA). Como parâmetros (*parameters*), esse nó recebeu um *schema* de entrada (*input_schema*) baseado nas informações do cliente (id, nome, endereço) e nos dados necessários para execução do processo, sendo essas informações armazenadas na *bag* do processo. Como *input*, foi fornecido ao nó dados de um cliente fictício e as informações necessárias para o menu de pizzas (pizza-menu) referentes ao pedido de pizza e às opções do formulário de pagamento (payment-form). O código referente a esse nó é apresentado no Trecho de Código 3:

Trecho de Código 3: trecho referente ao evento de início do processo

```
{
    "id": "START-PROCESS",
    "name": "Customer wants a pizza",
    "lane_id": "customer",
    "next": "ORDER-PIZZA",
    "type": "Start",
    "parameters": {
        "input_schema": {
            "type": "object",
            "properties": {
                "customer": {
                    "type": "object",
                    "properties": {
                        "id": { "type": "string", "format": "uuid" },
                        "name": { "type": "string" },
                        "address": { "type": "string" }
                    }
                },
                "pizza-menu": {
                    "type": "object",
                    "properties": {
                        "flavors": {
                            "type": "array",
                            "items": { "type": "string" }
                         }
                    }
                },
                "payment-form": {
                    "type": "object",
                    "properties": {
                        "options": {
                            "type": "array",
                                "items": { "type": "string" }
                        }
                    }
                }
            },
            "required": [
                            "customer.name", "customer.address", "customer.id",
                            "pizza-menu.flavors", "payment-form.options"
                        ]
        },
        "input": {
            "customer": {
                "id": "19d47eea-02e0-4f89-a59e-443503837c53",
                "name": "Dummy",
                "address": "Dummy avenue, 4135"
            },
            "pizza-menu": {
                "flavors": ["mozzarella", "pepperoni", "mushrooms", "sausage",
                            "green pepper", "onion", "extra cheese"]
            },
            "payment-form": {
                "options": ["cancel", "pay"]
            }
```

```
        }
    }
}
```

Todas as informações para a inicialização do processo foram marcadas como requeridas. O *next* desse nó foi setado para ORDER-PIZZA, a próxima tarefa a ser executada no processo.

Nó de pedido de pizza (ORDER-PIZZA)

O nó responsável pela tarefa de pedido da pizza por parte do cliente foi configurado como um nó de tarefa do usuário (*userTaskNode*). Esse nó possui um *schema* para descrever o formato esperado do *payload* de *commit* da tarefa (*activity_schema*) e o *activity_manager* foi setado para *commit*, fazendo com que o processo permaneça em estado de espera até que o *activity_manager* submeta a atividade para o processo. O Trecho de Código 4.1 apresenta o trecho mencionado:

```
{
    "id": "ORDER-PIZZA",
    "name": "Customer orders a pizza",
    "lane_id": "customer",
    "next": "TAKE-ORDER",
    "type": "UserTask",
    "parameters": {
        "action": "ORDER_PIZZA",
        "activity_manager": "commit",
        "activity_schema": {
            "type": "object",
            "properties": {
                "order": {
                    "type": "object",
                    "properties": {
                        "flavors": {
                            "type": "array",
                             "items": {
                                "type": "object",
                                "properties": {
                                    "qty": { "type": "integer" },
                                    "label": { "type": "string" }
                                }
                            }
                        }
                    }
                }
            }
        },
        …
```

Trecho de Código 4.1: trecho referente às informações gerais e *schema* da tarefa ORDER-PIZZA

O *schema* acima requer que o cliente informe, para cada sabor de pizza escolhido, a quantidade de pizzas daquele sabor e qual sabor é desejado. Nesse nó também foram configurados os dados de *input* da tarefa (bag.pizza-menu) e o formato esperado do resultado a ser utilizado no próximo nó (TAKE-ORDER). O resultado foi configurado para

conter, além dos dados requisitados no *schema* do *payload* da tarefa, informações como o identificador do pedido, a data do pedido e o nome e endereço do cliente, o que pode ser visto no Trecho de Código 4.2:

```
        "input": {
            "menu": {
                "$ref": "bag.pizza-menu"
            }
        },
        "result_schema": {
            "type": "object",
            "properties": {
                "order": {
                    "type": "object",
                    "properties": {
                        "order_id": { "type": "string" },
                       "date": { "type": "string", "format": "date" },
                        "customer_name": { "type": "string" },
                        "customer_address": { "type": "string" },
                        "flavors": {
                            "type": "array",
                            "items": {
                                "type": "object",
                                "properties": {
                                  "qty": { "type": "integer" },
                                  "label": { "type": "string" }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
```

Trecho de Código 4.2: trecho referente ao *input* e *results_schema* da tarefa ORDER-PIZZA

O *result* desse nó será utilizado na próxima tarefa, que irá encapsular esse *result* em um objeto juntamente com outras informações a serem armazenadas na *bag* do processo.

Nó de armazenamento do pedido na *bag* do processo (TAKE-ORDER)

Os dados do pedido serão necessários durante todo o processo, fazendo-se necessário armazenar esses dados na *bag* do processo para recuperá-los de acordo com a demanda. Para isso foi criado um nó de tarefa do sistema (*systemTaskNode*), responsável por resgatar os dados do *result* da tarefa anterior (ORDER-PIZZA) e armazená-los na *bag.* O Trecho de Código 5 apresenta a configuração na BP para esse nó:

```
{
    "id": "TAKE-ORDER",
    "name": "Systems takes the order from customers",
    "lane_id": "customer",
    "next": "PAY-ORDER",
    "type": "SystemTask",
```

```
    "category": "setToBag",
    "parameters": {
        "input": {
            "order": {
                "detail": { "$ref": "result.order" },
                "total": { "$js": "({result}) => { return
Math.round((result.order.flavors.map(x => x.qty).reduce((a,b) => a+b) * 10.15), 2) }" }
            }
        }
    }
}
```

Trecho de Código 5: trecho referente ao armazenamento de dados na *bag* do processo

A categoria do nó foi setada para *setToBag* e os dados do pedido encapsulados para armazenamento na *bag* do processo. Um trecho de código Java Script foi adicionado para calcular o valor total do pedido. Para exemplificação foi utilizado um valor fixo, independente do valor da pizza. No *next* desse nó (PAY-ORDER) o pedido será pago pelo cliente.

Nó de pagamento do pedido (PAY-ORDER)

Para pagamento do pedido foi configurada uma tarefa de usuário que recebe como *input* os dados do formulário de pagamento (bag.payment-form) e produz como resultado uma *string* referente ao status do pagamento do pedido, isto é, se o pedido foi cancelado (CANCELED), ocorreu um erro durante o pagamento do pedido (ERROR) ou se o pagamento foi realizado com sucesso (PAID). O Trecho de Código 6 apresenta a configuração para esse nó:

```
{
    "id": "PAY-ORDER",
    "name": "Customer pays the order or cancels it",
    "lane_id": "customer",
    "next": "ORDER-CANCELED",
    "type": "UserTask",
    "parameters": {
        "action": "PAY_ORDER",
        "activity_schema": {},
        "activity_manager": "commit",
        "input": {
            "form": {
                "$ref": "bag.payment-form"
            }
        },
        "result_schema": {
            "type": "object",
            "properties": { "payment_status": { "type": "string" } }
        }
    }
}
```

Trecho de Código 6: trecho referente à tarefa de pagamento do pedido

O próximo nó (ORDER-CANCELED) foi configurado para receber o resultado do nó de pagamento do pedido e realizar a operação de decisão necessária.

Nó de fluxo (ORDER-CANCELED) e nós finais correspondentes (END-WITHOUT-ORDER e END-WITH-ERROR)

De acordo com o diagrama BPMN apresentado na Figura 1, o nó de fluxo (*flowNode*) deve decidir entre três caminhos distintos. O caminho desejado, configurado como *default* no nó, representa o pagamento bem sucedido do pedido. O processo segue esse caminho quando o *result* do nó anterior emite a *string "*PAID". Nos outros dois caminhos o processo segue para nós finais para sua finalização, sendo um caminho devido ao cancelamento do pedido (emissão da *string* "CANCELED") e o outro caminho devido a um erro de pagamento (emissão da *string* "ERROR"). Essa informação está codificada na BP no *next* do nó de fluxo, como visto no Trecho de Código 7. O Trecho de Código 8 apresenta a codificação dos nós finais correspondentes:

```
{
    "id": "ORDER-CANCELED",
    "name": "Checks if the customer paid the order or if the order was canceled",
    "lane_id": "customer",
    "next": {
        "default": "SAVE-ORDER",
        "CANCELED": "END-WITHOUT-ORDER",
        "ERROR": "END-WITH-ERROR",
        "PAID": "SAVE-ORDER"
    },
   "type": "Flow",
   "parameters": {
       "input": {
           "decision": {
               "$ref": "result.payment_status"
           }
       }
   }
}
```

Trecho de Código 7: trecho referente ao nó de fluxo do processo

```
{
    "id": "END-WITHOUT-ORDER",
    "name": "Process finishes after the order's cancel",
    "lane_id": "customer",
    "next": null,
    "type": "Finish"
},
{
    "id": "END-WITH-ERROR",
    "name": "Process finishes after payment error",
    "lane_id": "customer",
    "next": null,
    "type": "Finish"
}
```

Trecho de Código 8: trecho referente aos nós finais correspondentes ao nó de fluxo do processo

Se a tarefa for bem sucedida, o próximo nó a ser executado é o nó de persistência do pedido no sistema (SAVE-ORDER).

Nó de persistência do pedido (SAVE-ORDER)

Para persistência do pedido no sistema foi configurado um nó de tarefa do sistema de categoria HTTP, utilizando-se um *endpoint* fictício. Nesse nó os dados do pedido são recuperados da *bag* do processo e um campo informando que o pedido está em fila para preparação é adicionado. Essa tarefa faz uma requisição POST enviando no corpo os dados do pedido com o status desejado. O Trecho de Código 9 apresenta a configuração correspondente desse nó na BP:

```
{
    "id": "SAVE-ORDER",
    "name": "System saves the order",
    "lane_id": "customer",
    "next": "PREPARE-ORDER",
    "type": "SystemTask",
    "category": "HTTP",
    "parameters": {
        "input": {
            "order": { "data": { "$ref": "bag.order" }, "status": "In Queue" }
        },
        "request": {
            "url": "https://pizzatie.com/v0/orders/save",
            "verb": "POST",
            "headers": {
                "ContentType": "application/json"
            }
        },
        "valid_response_codes": [200, 201]
    }
}
```

Trecho de Código 9: trecho referente ao nó de persistência do pedido no sistema

Com o pedido persistido, o pedido pode ser então preparado pelo restaurante (PREPARE-ORDER).

Nó de preparação do pedido (PREPARE-ORDER)

Configurado como um nó de tarefa do usuário, esse nó é responsável por recuperar os dados do pedido da *bag* do processo e emitir como resultado uma *string* informando que o pedido foi preparado. O Trecho de Código 10 mostra o código JSON correspondente:

```
{
    "id": "PREPARE-ORDER",
    "name": "Restaurant prepares an order",
    "lane_id": "restaurant",
    "next": "UPDATE-ORDER",
    "type": "UserTask",
    "parameters": {
        "action": "PREPARE_ORDER",
        "activity_manager": "commit",
        "activity_schema": {},
        "input": {
            "order": {
```

```
                "$ref": "bag.order.detail"
            }
        },
        "result_schema": { "type": "string" }
    }
}
```

Trecho de Código 10: trecho referente ao nó de preparação do pedido pelo restaurante

Após o pedido ser preparado seu status é atualizado no sistema (UPDATE-ORDER).

Nó de atualização do status do pedido (UPDATE-ORDER)

Para atualização do status do pedido no sistema foi configurado um nó de tarefa do sistema com a categoria HTTP, de modo semelhante ao nó TAKE-ORDER apresentado anteriormente. A única diferença desse nó é a recuperação apenas do identificador do pedido para correta atualização do status do mesmo, feito através de uma requisição POST para um *endpoint* fictício de atualização de status de pedidos. O Trecho de Código 11 apresenta o JSON referente a esse nó:

```
{
    "id": "UPDATE-ORDER",
    "name": "System updates the order status",
    "lane_id": "restaurant",
    "next": "DISPATCH-DELIVERY",
    "type": "SystemTask",
    "category": "HTTP",
    "parameters": {
        "input": {
            "order": {
                "order_id": { "$ref":"bag.order.detail.order_id" },
                "status": { "$ref": "result" }
            }
        },
        "request": {
            "url": "https://pizzatie.com/v0/orders/update",
            "verb": "POST",
            "headers": {
                "ContentType": "application/json"
            }
        },
        "valid_response_codes": [200, 201]
    }
}
```

Trecho de Código 11: trecho referente ao nó de atualização do status do pedido no sistema

Após atualização do status do pedido no sistema, o pedido é enviado para entrega pelo entregador (DISPATCH-DELIVERY).

Nó de entrega do pedido (DISPATCH-DELIVERY)

O nó de entrega do pedido foi configurado de maneira semelhante ou nó de preparação do pedido (PREPARE-ORDER). Esse nó referente a uma tarefa executada na

lane do entregador recebe como entrada os dados do pedido a ser entregue e informa a entrega do pedido, emitindo uma *string* para atualização do seu status no sistema (FINALIZE-ORDER), que é a próxima tarefa a ser executada. O Trecho de Código 12 apresenta o código referente a esse nó:

```
{
    "id": "DISPATCH-DELIVERY",
    "name": "Deliveryman gets an order to deliver",
    "lane_id": "deliveryman",
    "next": "FINALIZE-ORDER",
    "type": "UserTask",
    "parameters": {
        "action": "DISPATCH-DELIVERY",
        "activity_manager": "commit",
        "activity_schema": {},
        "input": {
            "order": {
                "$ref": "bag.order.detail"
            }
        },
        "result_schema": { "type": "string" }
    }
}
```

Trecho de Código 12: trecho referente à entrega do pedido pelo entregador

Nó de finalização do pedido no sistema

Após a entrega do pedido para o cliente, o pedido é finalizado no sistema. Para esse nó, foi utilizada uma configuração semelhante ao nó UPDATE-ORDER, onde simplesmente ocorre a atualização do status do pedido em um *endpoint* fictício utilizando o identificador do pedido armazenado na *bag* do processo. O Trecho de Código 13 apresenta o código correspondente a essa etapa:

```
{
                "id": "FINALIZE-ORDER",
                "name": "System finalizes order",
                "lane_id": "deliveryman",
                "next": "RATE-SERVICE",
                "type": "SystemTask",
                "category": "HTTP",
                "parameters": {
                    "input": {
                              "order": { "order_id": { "$ref":"bag.order.detail.order_id" },
"status": { "$ref": "result" } }
                    },
                    "request": {
                        "url": "https://pizzatie.com/v0/orders/update",
                        "verb": "POST",
                        "headers": {
                            "ContentType": "application/json"
                        }
                    },
                    "valid_response_codes": [200, 201]
                }
```

```
            }
```

Trecho de Código 13: trecho referente ao nó de finalização do pedido no sistema

Com a finalização do pedido no sistema, o cliente pode então classificar o serviço com base em sua experiência (RATE-SERVICE).

Nó de classificação do serviço (RATE-SERVICE)

Feita a entrega do pedido, o cliente pode realizar a classificação do serviço. A configuração desse nó de tarefa do usuário foi feita de maneira semelhante à configuração do nó ORDER-PIZZA. O nó RATE-SERVICE possui um *activity_schema* que requer o preenchimento da nota dada ao serviço e de um comentário por parte do cliente durante a execução da tarefa. Como saída (*result_schema*), além das informações de nota e comentário, o nó adiciona o identificador do pedido correspondente à classificação do serviço e o identificador do cliente. Esses dados são então passados adiante no processo. O Trecho de Código 14 mostra o JSON correspondente a esse nó:

```
{
    "id": "RATE-SERVICE",
    "name": "Customer rates the service after receiving the delivery",
    "lane_id": "customer",
    "next": "SAVE-RATING",
    "type": "UserTask",
    "parameters": {
        "action": "RATE_SERVICE",
        "activity_schema": {
            "type": "object",
            "properties": {
                "rating_form": {
                    "type": "object",
                    "properties": {
                        "rate": { "type": "integer" },
                        "comment": { "type": "string" }
                    }
                }
            }
        },
        "input": {},
        "result_schema": {
            "rating_form": {
                "type": "object",
                "properties": {
                    "order_id": { "type": "string" },
                    "customer_id": { "type": "string" },
                    "rate": { "type": "integer" },
                    "comment": { "type": "string" }
                }
            }
        }
    }
}
```

Trecho de Código 14: trecho referente ao nó de classificação do serviço pelo cliente

Com as informações requisitadas preenchidas, a classificação do cliente pode ser persistida no sistema (SAVE-RATING).

Nó de persistência da classificação do serviço (SAVE-RATING)

Com o *result* proveniente do nó anterior, os dados de classificação do serviço feita pelo cliente podem ser salvos. O nó SAVE-RATING foi configurado de modo semelhante ao nó SAVE-ORDER. As únicas diferenças são os dados sendo persistidos e o *endpoint* sendo utilizado na requisição POST da tarefa de sistema. O Trecho de Código 15 apresenta o código correspondente a esse nó:

```
{
    "id": "SAVE-RATING",
    "name": "System saves the rating",
    "lane_id": "customer",
    "next": "END-WITH-RATING",
    "type": "SystemTask",
    "category": "HTTP",
        "parameters": {
            "input": {
                "rating": { "$ref": "result.rating_form" }
            },
            "request": {
                "url": "https://pizzatie.com/v0/service/ratings",
                "verb": "POST",
                "headers": {
                    "ContentType": "application/json"
                 }
            },
            "valid_response_codes": [200, 201]
    }
}
```

Trecho de Código 15: trecho referente ao nó de persistência da classificação do serviço no sistema

Persistida a classificação do cliente no sistema, o processo segue para seu nó final (END-WITH-RATING).

Nó final do processo (END-WITH-RATING)

Por fim, o Trecho de Código 16 apresenta o código referente ao evento de finalização do processo, disparado após os dados da classificação do serviço serem persistidos no sistema:

```
{
    "id": "END-WITH-RATING",
    "name": "Process finishes after the customer's rating",
    "lane_id": "customer",
    "next": null,
    "type": "Finish"
}
```

Trecho de Código 16: trecho referente ao nó final do processo

Conceitos cobertos com a solução e realização do desafio

Com a solução apresentada acima foi possível cobrir os seguintes tipos de nó utilizados no *flowbuild* para construção da BP, presentes na Tabela 1:

Tipo de nó	Categoria	Nós	Quantidade
startNode	-	CUSTOMER WANTS A PIZZA	1
userTaskNode	-	ORDER PIZZA, PAY ORDER, PREPARE ORDER, DISPATCH DELIVERY e RATE SERVICE	5
systemTaskNode	SetToBag	TAKE ORDER	1
	HTTP	SAVE ORDER, UPDATE ORDER, FINALIZE ORDER e SAVE RATING	4
flowNode	-	ORDER CANCELED?	1
finishNode	-	END WITHOUT ORDER, END WITH ERROR e END WITH RATING	3
Total			15

Outros conceitos referentes a informações necessárias a cada tipo de nó podem ser visualizados na BP apresentada anteriormente.