sistema PizzaNet, no qual podemos perceber que este é composto de dois subsistemas, o subsistema de vendas e o subsistema administrativo. Podemos perceber também que o subsistema administrativo tem uma dependência com relação ao subsistema de vendas, uma vez que é necessário haver pedidos para que os módulos administrativos possuam dados que justifiquem sua execução.

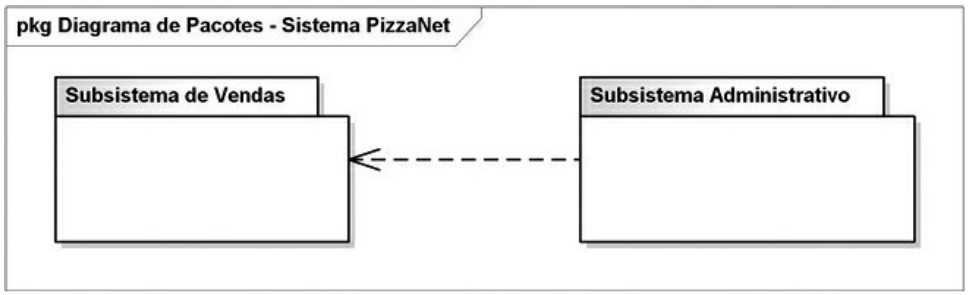

Figura 17.7 – Diagrama de Pacotes PizzaNet.

17.2.6 Diagramas de Sequência da PizzaNet

Nesta seção, apresentaremos os diagramas de sequência referentes a esse estudo de caso, que também são uma forma de documentar cada caso de uso apresentado no início deste capítulo. Talvez alguns dos métodos apresentados nos diagramas a seguir pudessem ser simplificados e algumas vezes poderia não ser necessária a chamada de outros métodos para complementá-los, podendo tais métodos ser absorvidos em alguns casos pelos métodos que os chamaram. No entanto, optamos por uma abordagem mais detalhada para facilitar a compreensão dos diagramas, mesmo porque os diagramas não estão baseados em nenhuma linguagem de programação específica, o que, aliás, é o correto.

Além disso, como foi dito anteriormente, a rigor deve haver um método `get` e um método `set` para cada atributo de uma classe, mas isso é impraticável em processos que contenham um número grande de atributos, além de desnecessário e mesmo não recomendado, uma vez que os diagramas são modelos de nível mais alto. Assim, utilizamos algumas vezes métodos como "registrar", para instanciar um objeto, ou "consultar", para retornar todas as informações de uma instância.

Diagrama de Sequência Escolher Pizza

A figura 17.8 apresenta o detalhamento do processo **Escolher Pizza** por

meio de um diagrama de sequência.

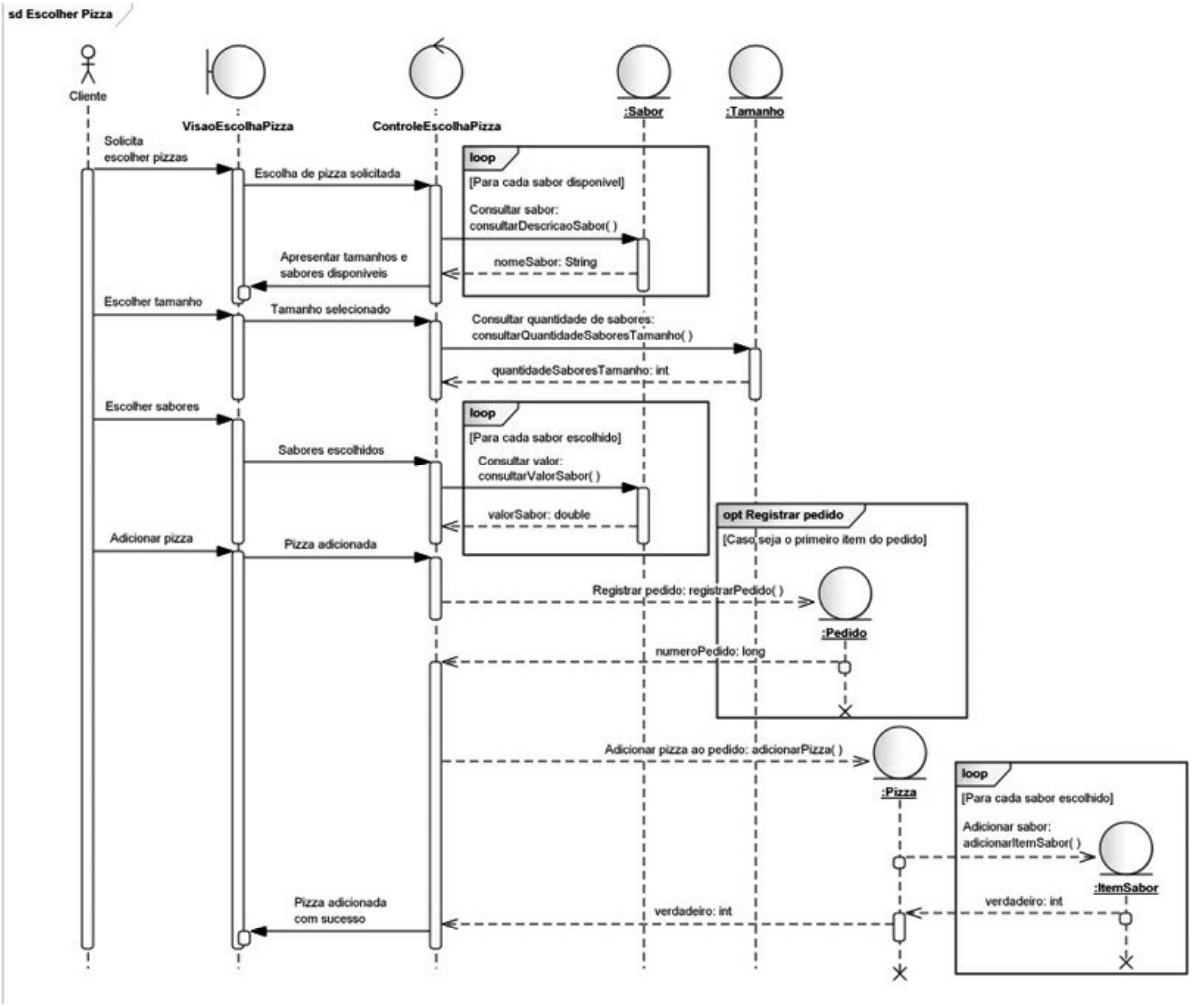

Figura 17.8 – Diagrama de Sequência Escolher Pizza.

Esse processo inicia-se quando o cliente acessa a página da PizzaNet, por esta ser a opção default; ou quando clica o botão Pizzas da interface. Ao ser avisado da ocorrência desse evento, o controlador dispara o método `consultarDescricaoSabor` em cada objeto da classe `Sabor`, para retornar sua descrição. Observe que existe um fragmento combinado do tipo `loop`, que demonstra que isso ocorre por meio de um laço. A partir das descrições retornadas pelas chamadas desse método, o controlador manda carregar na página os tamanhos e os sabores disponíveis.

O cliente deve, então, selecionar o tamanho desejado da pizza, que é repassado para o controlador que dispara o método `consultarQuantidadeSaboresTamanho` para retornar a quantidade de

sabores que podem ser pedidos para o tamanho escolhido. Depois disso, o cliente poderá escolher os sabores desejados para a pizza, simplesmente clicando sobre o sabor. A escolha desses sabores é repassada para o controlador que executará um laço, disparando o método `consultarValorSabor` para cada sabor escolhido, recuperando o valor do sabor. Quando o cliente tiver terminado de escolher os sabores da pizza, bastará pressionar o botão adicionar. Esse evento obrigará o controlador a realizar um teste para determinar se a pizza escolhida é o primeiro item do pedido, conforme demonstra o fragmento combinado do tipo **opt**. Caso o teste seja positivo, o controlador disparará o método `registrarPedido` que instanciará um novo objeto da classe `Pedido` e retornará o número deste.

O controlador, então, executará o método `adicionarPizza` para adicionar a nova pizza ao pedido. Este é um método construtor, que instanciará um novo objeto da classe `Pizza` e chamará o método `adicionarItemSabor` para instanciar os objetos da classe `ItemSabor` associados à pizza. O método será disparado tantas vezes quantos forem os sabores escolhidos, como demonstra o fragmento combinado do tipo loop. Esse método é também um método construtor. Se o retorno desses métodos for verdadeiro, será apresentada a mensagem de "Pizza adicionada com sucesso".

Diagrama de Sequência Escolher Bebida

Esse processo se inicia quando o cliente pressiona o botão **Bebidas** na página da PizzaNet. Esse evento é repassado pela página à controladora que ordena à página que seja apresentado o formulário para escolha de bebidas, contendo os tipos de bebidas oferecidos pela pizzaria. Atualmente, os tipos são suco, refrigerante e cerveja, mas, futuramente, poderão ser acrescentados novos tipos de bebidas. Por esse motivo, todos os objetos da classe `TipoBebida` são recuperados e apresentados (Figura 17.9).

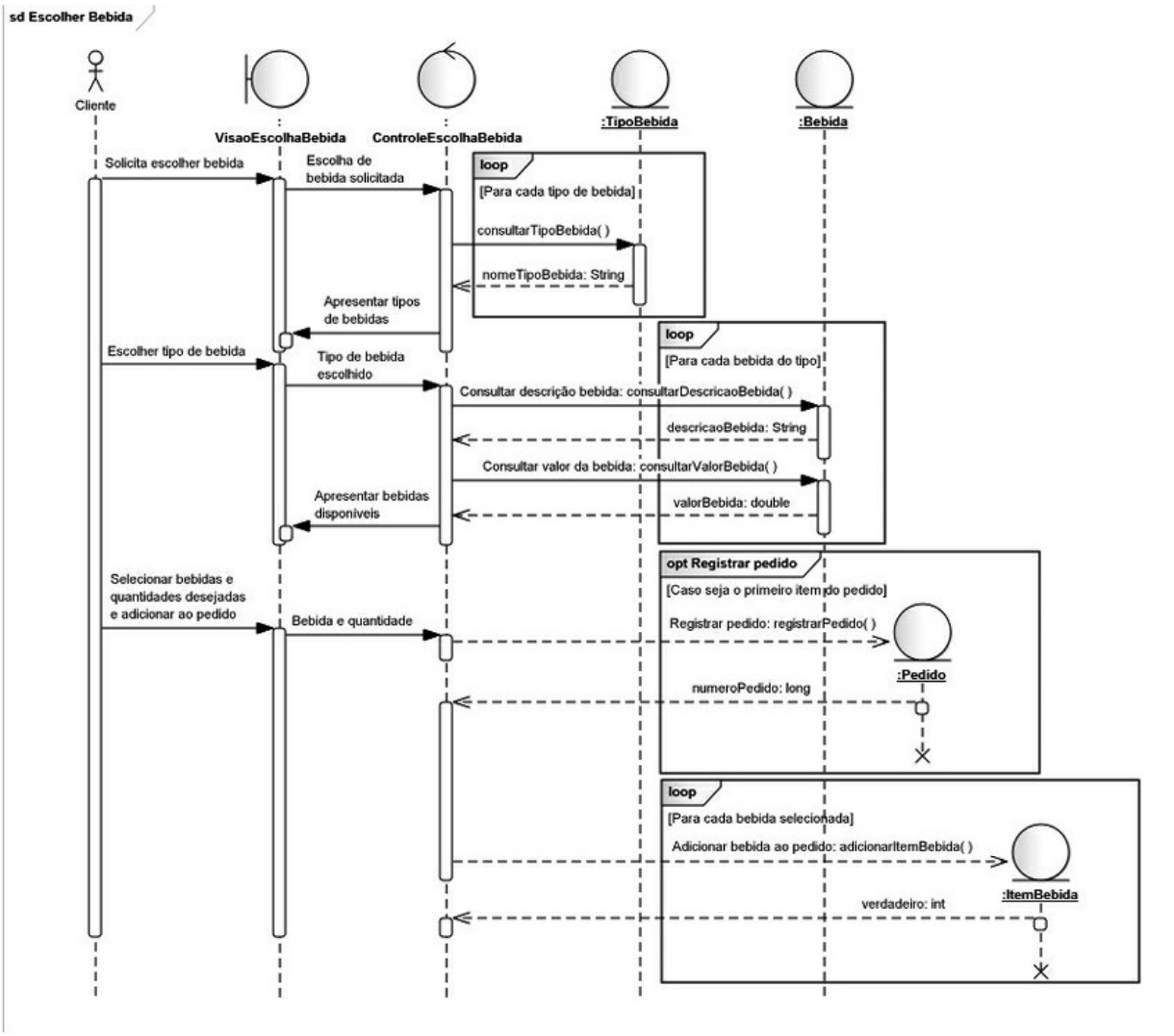

Figura 17.9 – Diagrama de Sequência Escolher Bebida.

Nesse formulário, o cliente deve escolher o tipo de bebida que deseja. Esse evento faz o controlador consultar todas as bebidas, com seus respectivos valores, do tipo selecionado. Isso é feito por meio de um laço, representado pelo fragmento combinado do tipo **loop**, no qual são executados os métodos `consultarDescricaoBebida` e `consultarValorBebida` para retornar a descrição e o valor de cada bebida do tipo escolhido. Com essas informações, o controlador mandará atualizar a página, apresentando todas as bebidas disponíveis ao cliente.

Em seguida, o cliente deverá selecionar a bebida e a quantidade desejada e pressionar o botão **Adicionar**, para acrescentar a bebida ao pedido. Esse evento fará o controlador realizar novamente o teste descrito no processo

anterior, para determinar se este é o primeiro item do pedido do cliente, conforme demonstra o fragmento combinado do tipo **loop**. Caso isso seja verdadeiro, o controlador disparará o método `registrarPedido` em um objeto da classe `Pedido`, instanciando-o e retornando o número do novo pedido.

Após esse teste, será executado outro laço, no qual o controlador chamará o método `adicionarItemBebida` para instanciar um novo objeto da classe `ItemBebida`, relativo a cada bebida que foi adicionada ao pedido. O retorno verdadeiro (na verdade, o valor 1) significará que a bebida em questão foi adicionada com sucesso.

Diagrama de Sequência Logar

Esse processo pode ser chamado pelo cliente quando este clicar o botão **Logar** ou pode ser chamado pelo processo **Concluir Pedido**, caso o cliente não tenha se autenticado ainda. A chamada a esse processo faz a controladora solicitar a apresentação do formulário de login na página da PizzaNet (Figura 17.10).

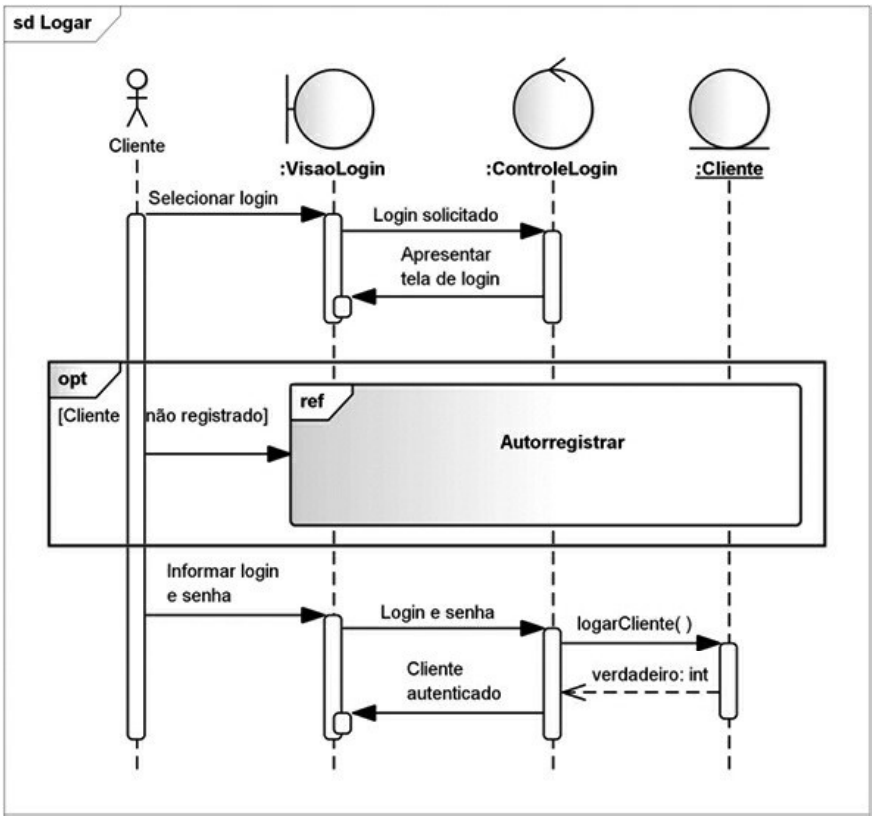

Figura 17.10 – Diagrama de Sequência Logar.

Caso o cliente não esteja cadastrado no sistema, ele terá a opção de se autorregistrar, conforme é demonstrado pelo fragmento combinado do tipo **opt**, que envolve o comportamento que será executado caso o cliente não esteja registrado ainda no sistema, ou seja, o processo de **Autorregistrar**, que será chamado pelo uso de interação que referencia esse processo.

Se já estiver registrado, o cliente informará, então, seu login e senha, fazendo o controlador disparar o método `logarCliente`. O retorno do valor verdadeiro para o controlador fará que este ordene a apresentação da mensagem "Cliente autenticado".

Diagrama de Sequência Autorregistrar

Este é um processo bastante simples. No momento em que o cliente solicita a execução desse processo, a controladora, em resposta, solicita a apresentação do formulário de registro, onde o cliente deverá inserir os seus dados e confirmar. Ao receber a confirmação, o controlador disparará o método `registrarCliente` e instanciará um novo objeto da classe `Cliente (`Figura 17.11)..

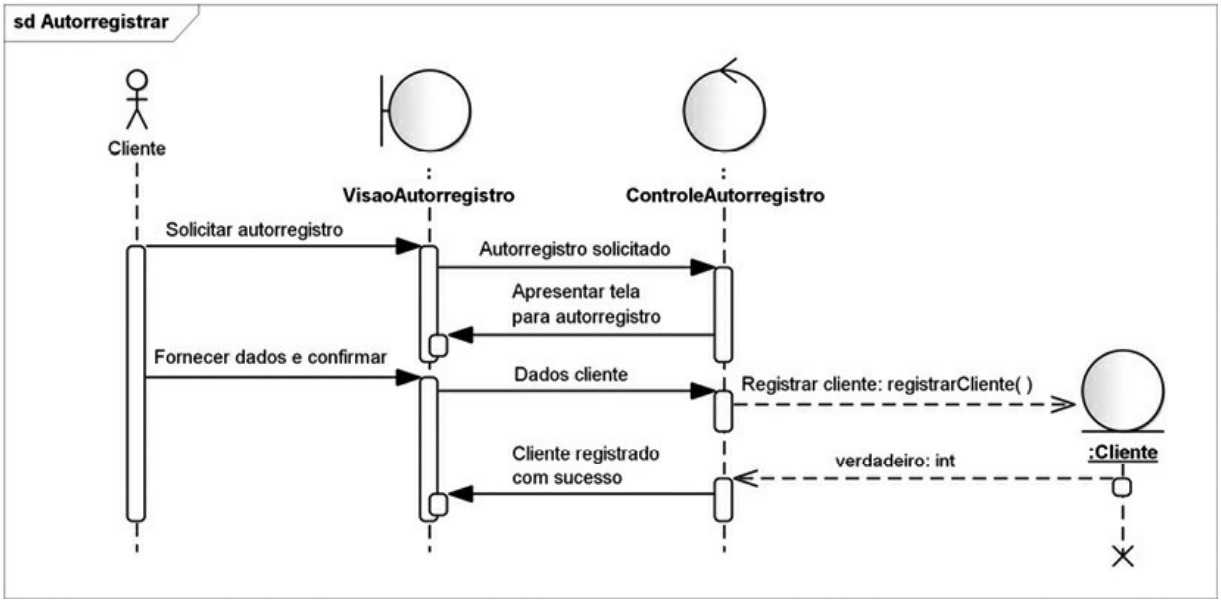

Figura 17.11 – Diagrama de Sequência Autorregistrar.

Diagrama de Sequência Opinar

Este é também um processo muito simples, em que, ao receber a solicitação de execução dessa funcionalidade, a controladora ordena que

seja apresentado o formulário para registro de opiniões. Ao receber a opinião do cliente, a controladora executará o método `registrarOpiniao`, para instanciar um novo objeto da classe `Opiniao`, e o processo estará finalizado (Figura 17.12).

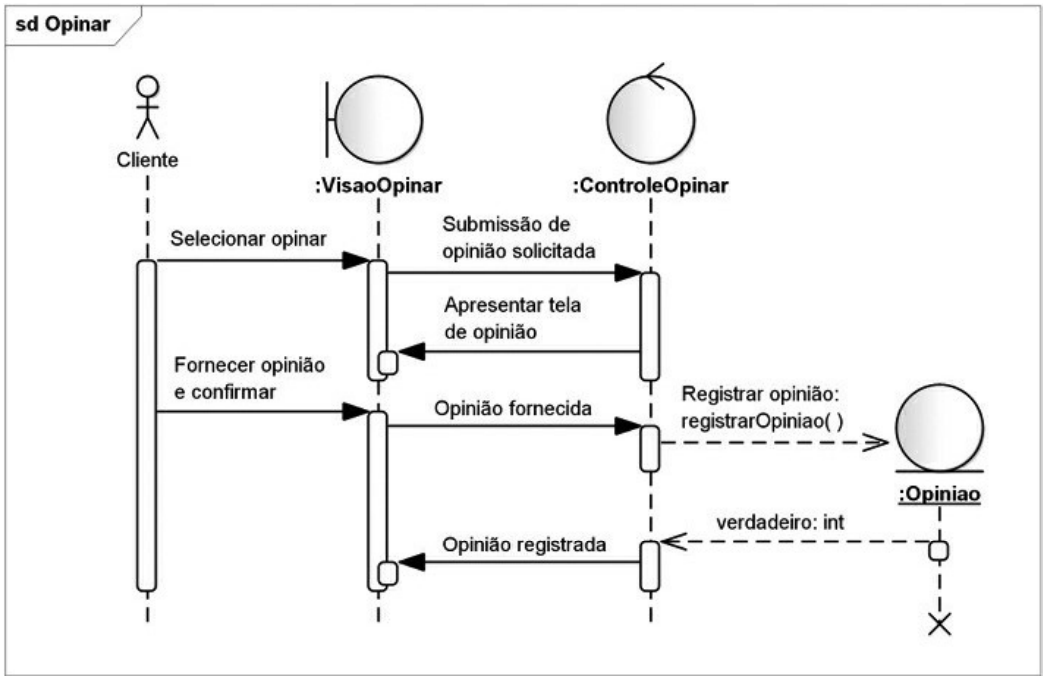

Figura 17.12 – Diagrama de Sequência Opinar.

Diagrama de Sequência Visualizar Pedido

Quando o cliente solicita a visualização dos itens de seu pedido, o controlador dispara o método `consultarPedido` em um objeto da classe `Pedido`, para retornar as informações do pedido feito pelo cliente, ou seja, as pizzas e bebidas solicitadas.

Como os objetos da classe `Pedido` precisam que suas informações sejam complementadas por objetos da classe `Pizza` e `ItemBebida`, o método `consultarPedido` dispara o método `consultarPizza` para consultar cada pizza associada ao pedido, como demonstra o fragmento combinado do tipo **loop**. Por sua vez, esse método dispara o método `consultarDescricaoTamanho` em um objeto da classe `Tamanho`, para retornar o tamanho da pizza consultada e, em seguida, chama o método `consultarItemSabor` em cada objeto da classe `ItemSabor` associado à pizza, como demonstra o segundo fragmento combinado do tipo **loop**,

uma vez que um objeto da classe `Pizza` também precisa que suas informações sejam complementadas pelos objetos dessa classe. Ainda dentro desse laço, o método `consultarItemSabor` chama o método `consultarDescricaoSabor` para retornar a descrição do sabor relacionado ao objeto `ItemSabor` em cada iteração do laço. Os resultados da consulta de cada pizza são retornados na forma de uma `String` para o método `consultarPedido` (Figura 17.13).

O método `consultarPedido`, então, inicia um segundo laço, demonstrado por mais um fragmento combinado do tipo **loop**, disparando o método `consultarItemBebida` em cada objeto da classe `ItemBebida` relacionado ao pedido. Por sua vez, esse método chama o método `consultarDescricaoBebida` para retornar a descrição da bebida. As informações de cada bebida são, então, retornadas ao método `consultarPedido` que, em posse de todas essas informações, retorna-as ao controlador que as manda apresentar na página da PizzaNet.

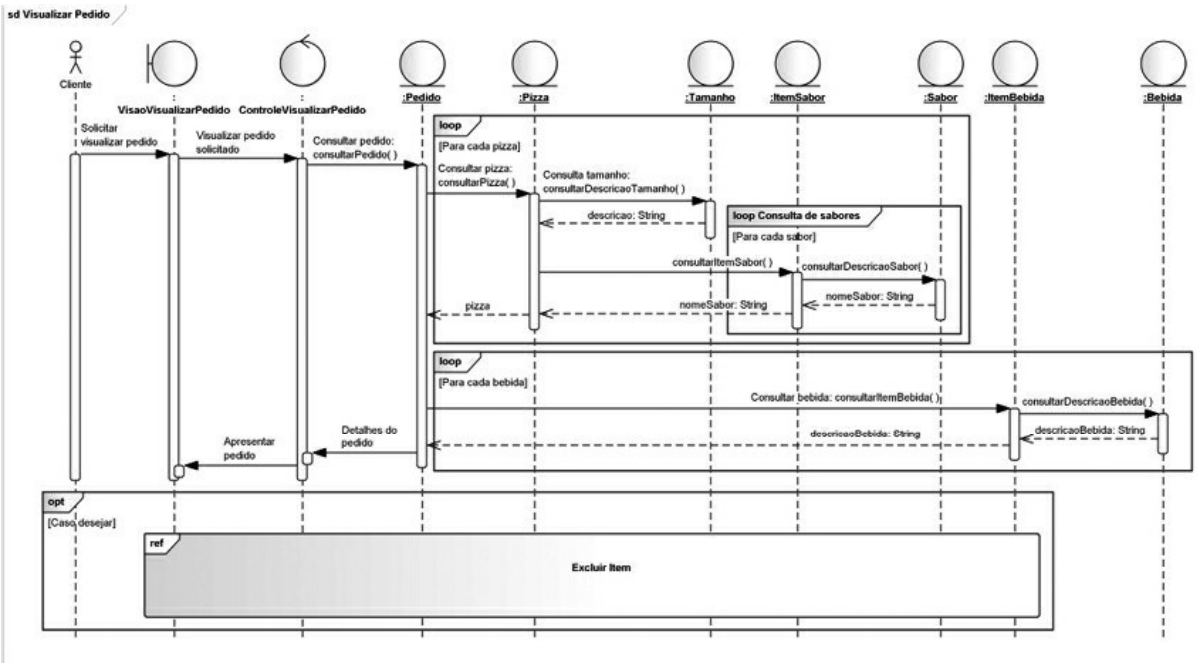

Figura 17.13 – Diagrama de Sequência Visualizar Pedido.

A partir dessa listagem, o cliente pode excluir qualquer um dos itens apresentados. Como isso é opcional, esse comportamento é apresentado por meio de um fragmento combinado do tipo **opt**, que contém um uso de interação que chama o processo de **Excluir Item**.

Diagrama de Sequência Excluir Item

Esse processo é chamado a partir do processo anterior, e quando se inicia, a controladora precisa escolher entre dois comportamentos possíveis, como demonstra o fragmento combinado do tipo **alt**.

Assim, se o item a excluir for uma pizza, a controladora dispara o método `excluirPizza` para excluir a pizza em questão. Esse método, por sua vez, dispara o método `excluirItemSabor` em cada objeto da classe `ItemSabor` associado à pizza. Isso é necessário porque existe uma associação de composição entre a classe `Pizza` e a classe `ItemSabor` e, por esse motivo, quando um objeto da classe `Pizza` for excluído, devem ser excluídos também todos os objetos `ItemSabor` a ele associados. Observe que esses dois métodos são destrutores, como demonstra o "`X`" que interrompe a linha de vida dos objetos (Figura 17.14).

Já se o item a excluir for uma bebida, o controlador chamará o método `excluirItemBebida` para destruir o objeto da classe `ItemBebida`. Este também é um método destrutor, como também demonstra o "`X`" que interrompe a linha de vida do objeto da classe `ItemBebida`.

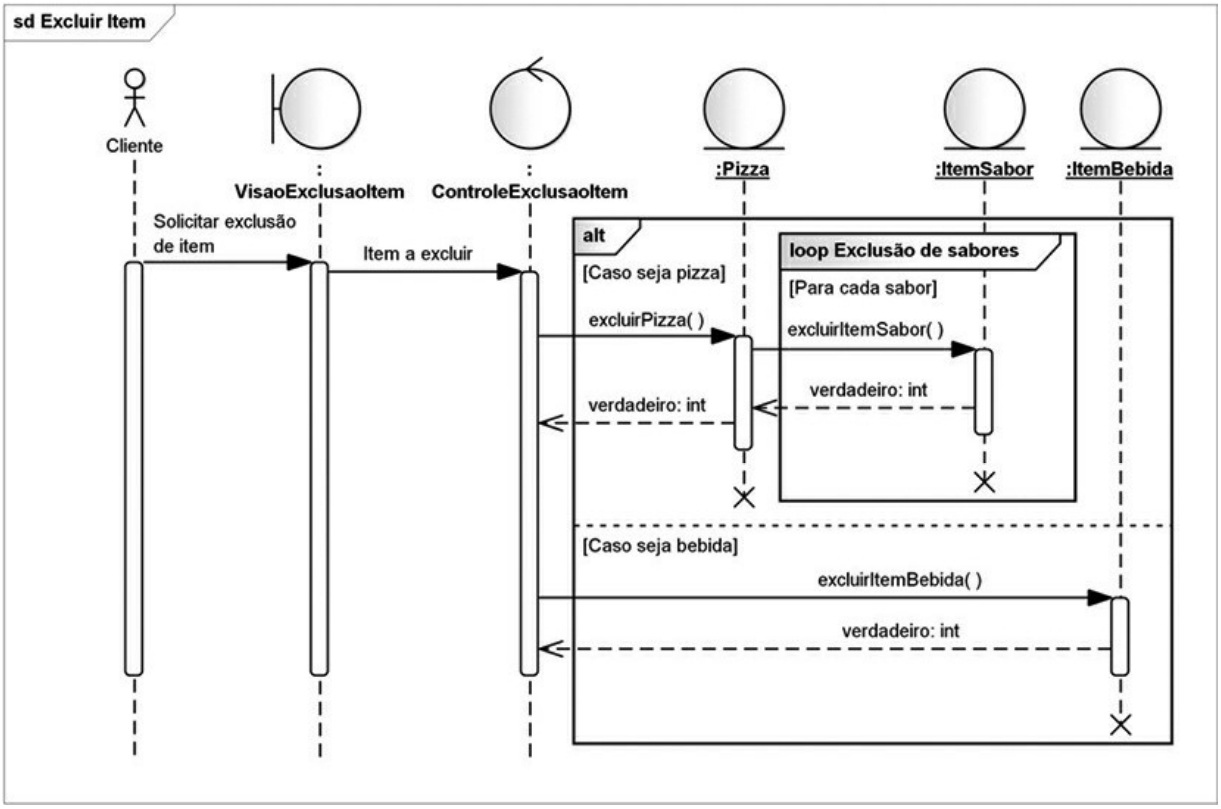

Figura 17.14 – Diagrama de Sequência Excluir Item.

Diagrama de Sequência Visualizar Pedidos Anteriores

Essa funcionalidade só pode ser solicitada se o cliente estiver autenticado. Quando o cliente seleciona essa opção, a controladora entra em um laço, demonstrado por um fragmento combinado do tipo **loop**, em que disparará o método `visualizarPedidosAbertos` para cada objeto da classe `Pedido` associado ao cliente. Esse método retornará o número de um pedido e, com esse número, chama-se o processo **Visualizar Pedido** aqui referenciado por um uso de interação, que apresentará os detalhes do pedido (Figura 17.15).

A partir da listagem dos pedidos anteriores, o cliente tem a opção de solicitar um pedido novamente. Como isso é opcional, esse comportamento é representado por um fragmento combinado do tipo **opt**. Nesse caso, o cliente deverá selecionar um pedido, o que fará a controladora disparar o método `registrarPedido` para instanciar um novo pedido.

Após obter o número do novo pedido, a controladora dispara o método `adicionarPizza` para instanciar um novo objeto da classe `Pizza.` Isso é feito para cada pizza contida no pedido anterior escolhido, como demonstra o fragmento combinado do tipo **loop**. Por sua vez, esse método inicia um novo laço representado por um segundo fragmento combinado do tipo **loop**, disparando o método `adicionarItemSabor` para instanciar um novo objeto da classe `ItemSabor.` Esse laço é executado tantas vezes quantos forem os sabores da pizza.

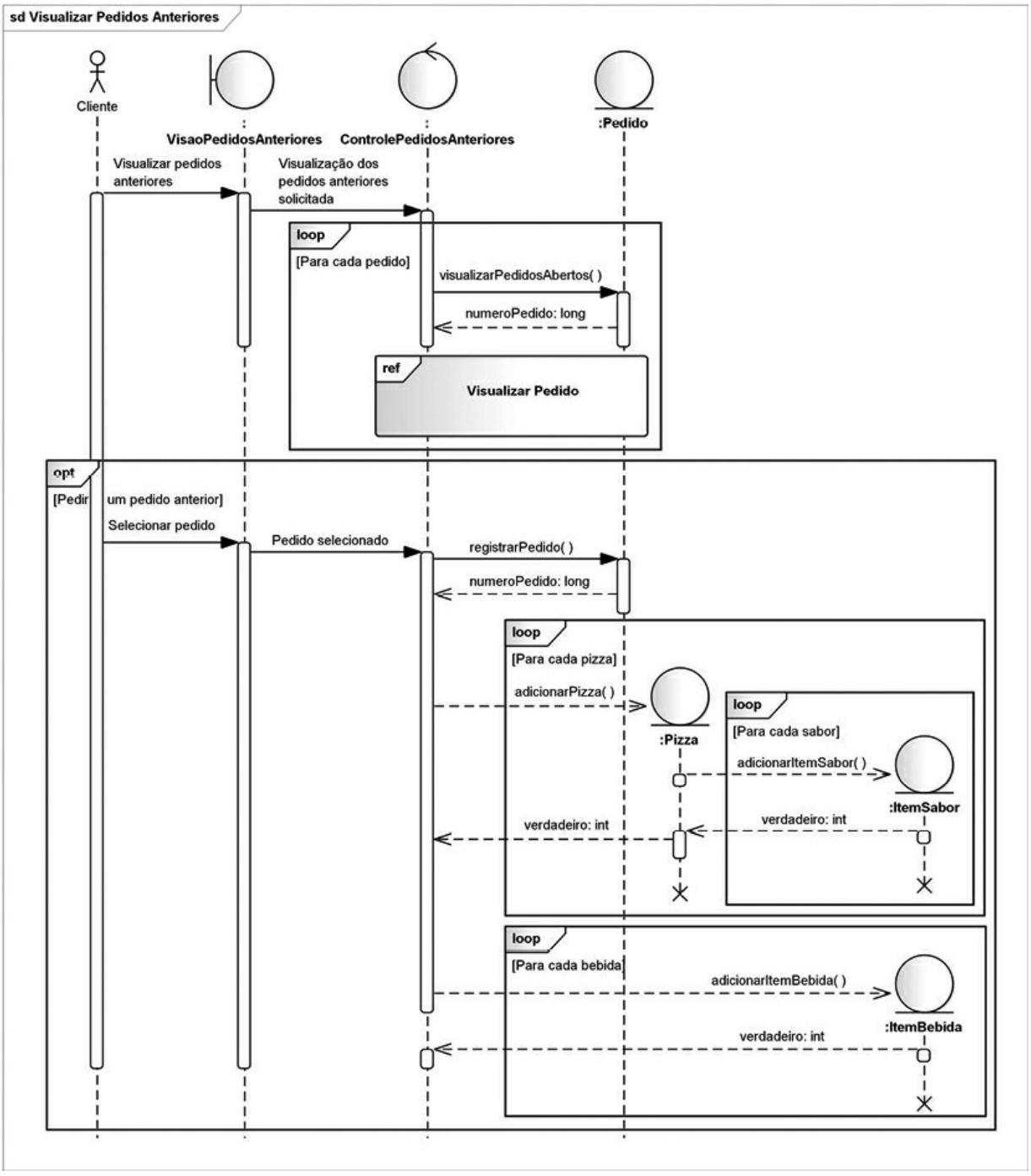

Figura 17.15 – Diagrama de Sequência Visualizar Pedidos Anteriores.

Após instanciar as pizzas do novo pedido, a controladora inicia um novo laço, representado por mais um fragmento combinado do tipo loop, onde é disparado o método `adicionarItemBebida`, instanciando um objeto da classe `ItemBebida` para cada bebida contida no pedido anterior.

Diagrama de Sequência Visualizar Sabores Mais Pedidos

Nesse processo, a controladora executa um laço representado por um fragmento combinado do tipo **loop**, onde é chamado o método `totalizarSabor.` Esse método soma todos os objetos da classe `ItemSabor` associados a um determinado sabor, retornando seu total quando o sabor muda e reiniciando a contagem (Figura 17.16).

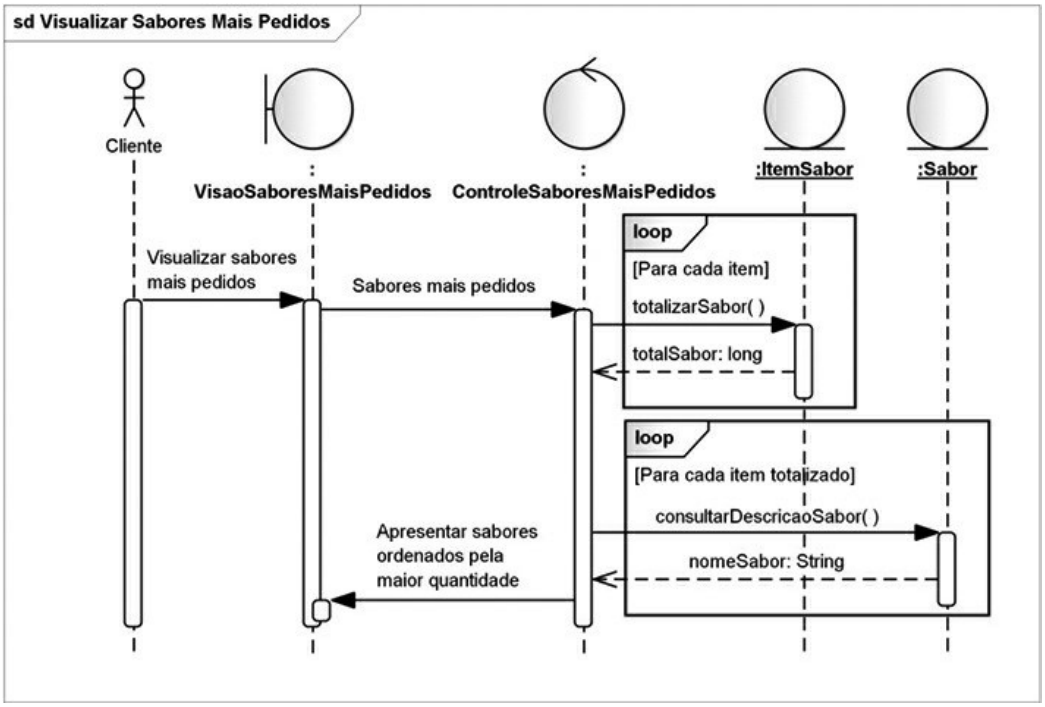

Figura 17.16 – Diagrama de Sequência Visualizar Sabores Mais Pedidos.

Ao receber esses totais, a controladora inicia outro laço para consultar a descrição de cada sabor totalizado, por meio do método `consultarDescricaoSabor`, que retorna uma `String` contendo a descrição do sabor consultado.

Diagrama de Sequência Concluir Pedido

Ao receber a solicitação de conclusão do pedido do cliente, a controladora primeiramente necessita verificar se o cliente já se encontra autenticado, conforme demonstra o fragmento combinado do tipo **opt**. Caso o cliente ainda não tenha se logado, é necessário executar o processo de **Logar**, referenciado por meio do uso de interação contida dentro do fragmento (Figura 17.17).

A seguir, é necessário executar o processo de **Visualizar Pedido**,