Universidade do Minho

Conselho de Cursos de Engenharia

Licenciatura em Engenharia Informática

3ºAno

Disciplina de Desenvolvimento de Sistemas de Software

Ano Lectivo de 2009/2010

Relatório do projecto – 2º Fase

Grupo 30

Dezembro, 2009

Identificação do grupo:

Grupo 30

Nome: Ivo Madeira Macedo

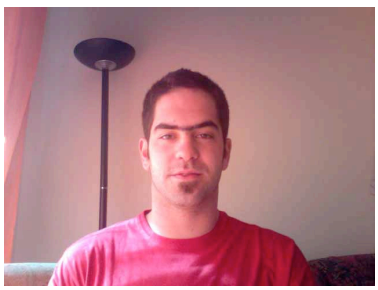
Número: 47120

Fotografia:

Nome: José Filipe Figueiredo

Número: 47046

Fotografia:



Resumo

Encontramo-nos na segunda parte do projecto da disciplina de Desenvolvimentos de Sistemas de Software – a entrega da arquitectura (diagrama de classes)e comportamento e interacção entre componentes (diagrama de sequencia para as interacções descritas nos use cases) do sistema proposto para satisfazer os requisitos identificados na primeira parte do projecto.

Área de Aplicação: Modelação e Desenvolvimento de Sistemas de Software

Palavras-Chave: Unified Modeling Language , Diagramas de Sequencia .

Índice:

1. Introdução:

No âmbito da disciplina de Desenvolvimento de Sistemas de Software da Licenciatura em Engenharia Informática da universidade do Minho, na primeira fase do projecto realizamos o levantamento de requisitos para a nossa aplicação, concentrando-nos agora na estrutura da nossa aplicação, isto é, o diagrama de classes final, e os diagramas de sequencia.

Os diagramas de sequencia representam o comportamento dos métodos dos objectos que estão na camada **Business** ou **Aplication Logic.** As mensagens vindas do actor representam as mensagens vindas da camada **Interface** para a camada **Business.** Os métodos invocados na camada **Business** podem devolver mensagens, para além das mensagens de resultado, do tipo **int, bool** ou **String**, as quais vão ser interpretadas na camada interface como mensagens de sucesso ou insucesso.

2. Diagramas de Sequência:

Os Diagrama de sequência (ou Diagrama de Sequência de Mensagens) são diagramas usados em UML (Unified Modeling Language), onde representam a sequência de processos numa aplicação ao longo do tempo, isto é, a troca de mensagens entre objectos.

Num projecto de construção de software, há uma grande quantidade de métodos em classes diferentes, no qual é complicado determinar a sequência global do comportamento do sistema, o diagrama de sequência representa essa informação de uma forma simples e lógica.

Estes diagramas são construídos a partir dos Use Case´s, definindo-se inicialmente qual o papel do sistema (Use Cases), e posteriormente é definido como o software realizará seu papel (Sequência de operações).

O diagrama de sequência realça a ordenação temporal em que as mensagens são trocadas entre os objectos do sistema.

Entende-se por mensagens os serviços solicitados de um objecto a outro, e as respostas desenvolvidas para as solicitações.

Conceitos:

Actores: São entidades externas que interagem com o sistema e que solicitam serviços, gerando dessa forma eventos que iniciam processos.

Objectos: Representam as instâncias das classes representadas no processo sendo representados como rectângulos.

Fragmento: Fragmentos de interacção tais como Alt (Alternativa), Opt (Opcional), Break (Parar), Loop (Repetição) entre outros.

Linha de vida: As linhas de vida representam a sequencia de mensagens do objecto durante a interacção representada, ao longo do tempo.

Subsistema Finanças

Todas as funcionalidades representadas pelos diagramas de sequência seguintes são da responsabilidade do subsistema finanças.

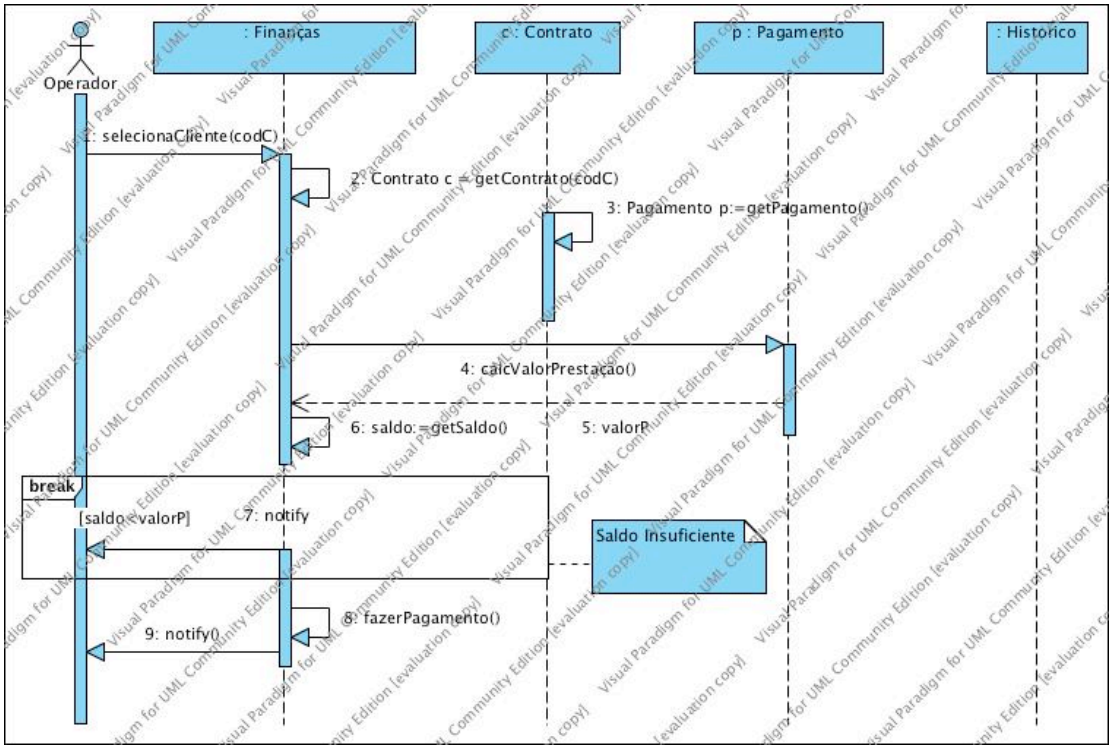


Diagrama de Sequência 1: Fazer Pagamento

Neste diagrama é representado o comportamento do sistema aquando é efectuado um pagamento á GereComSaber por parte de um Cliente.

Como é facilmente observável neste diagrama, para um pagamento ser efectuado há uma primeira interacção do Actor (Operador) com o sistema introduzindo o código de cliente em questão. Já no subsistema Finanças, para conhecer o montante a pagar, é necessária uma série de interacções a vários objectos uma vez que o valor da prestação se encontra representada entidade Pagamento.

Só depois desta série de interacções necessárias para conhecer o valor da prestação a pagar é que é feito o teste (verificar se o saldo é suficiente) representado pelo frame **break ,** resultando numa excepção caso o saldo não seja suficiente.

Subsistema GestãoC

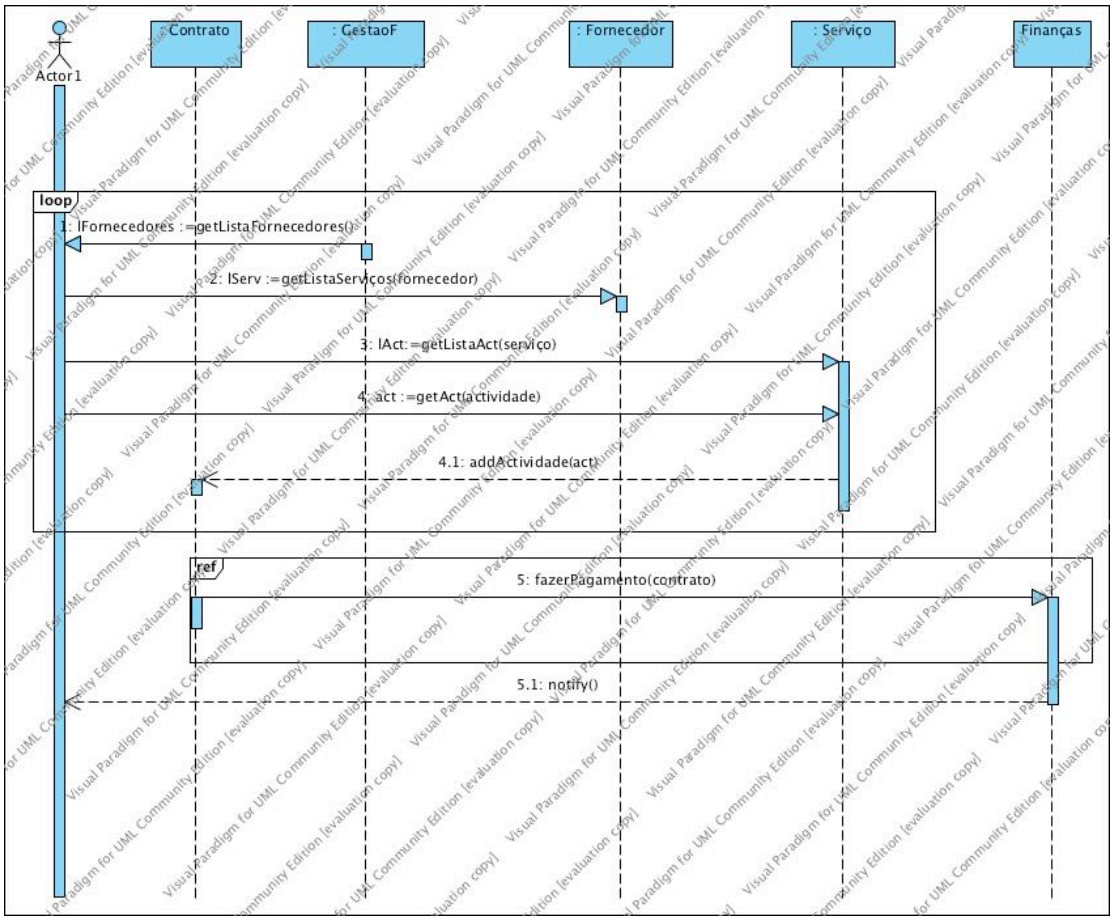


Diagrama de Sequência 2 : Adicionar novo Contrato

Este diagrama retrata o comportamento da funcionalidade de adicionar um novo contrato á GereComSaber.

Como se pode verificar é fornecida ao actor (Operador) a listagem de todos os serviços e actividades dos fornecedores. São adicionadas todas as Actividades seleccionadas pelo Operador - frame **loop** garante que o Operador se encontra a escolher serviços até ele próprio confirmar o término da adição dos mesmos.

Finalmente é gerado o Contrato e guardado nas Finanças.

Neste diagrama é representada a funcionalidade de remover actividades do contrato do Cliente designado pelo Operador.

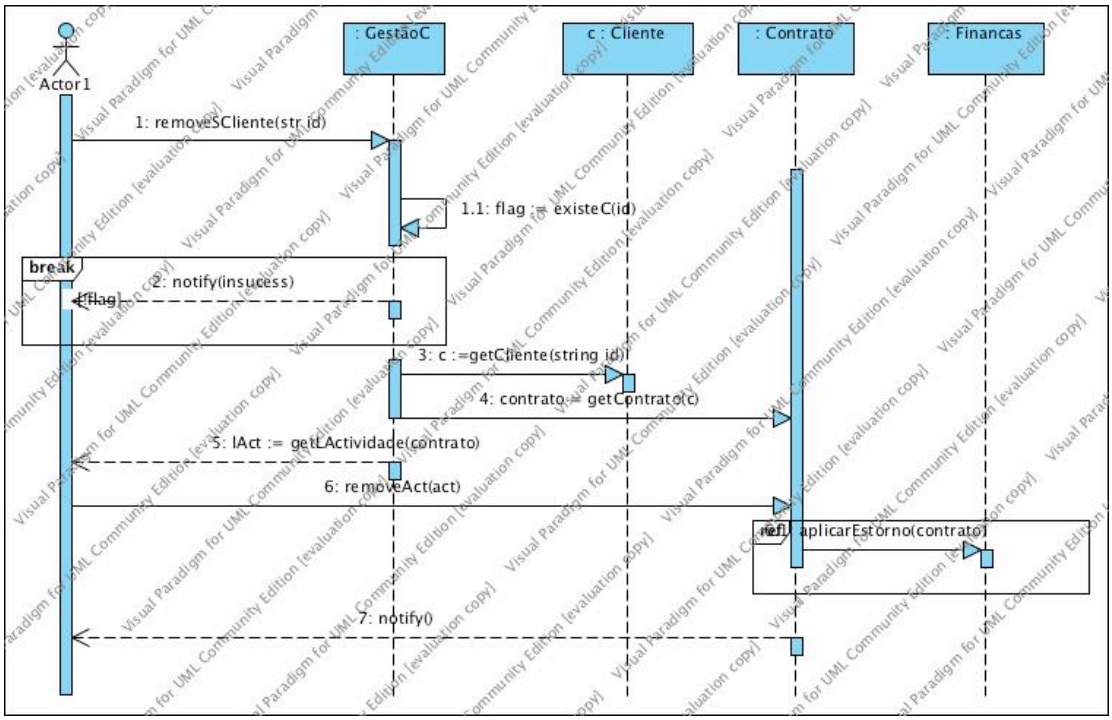


Diagrama de Sequência 3 : Remover Serviço Cliente

O operador designa o Cliente em questão, é feito um teste na entidade GestaoC de modo a verificar se o Cliente existe no sistema. Caso o Cliente não exista é gerada um excepção (frame **break**) .

Caso exista é disponibilizada a lista de actividades presentes no contrato, o Operador designa as actividades a remover do seu contrato. As actividades são removidas do contrato e é imediatamente invocado o comportamento do diagrama : **Aplicar Estorno** , que tem a responsabilidade de actualizar o valor da prestação a partir daí e devolver o valor do estorno ao **Cliente,** tal como podemos observar no respectivo diagrama de sequencia .

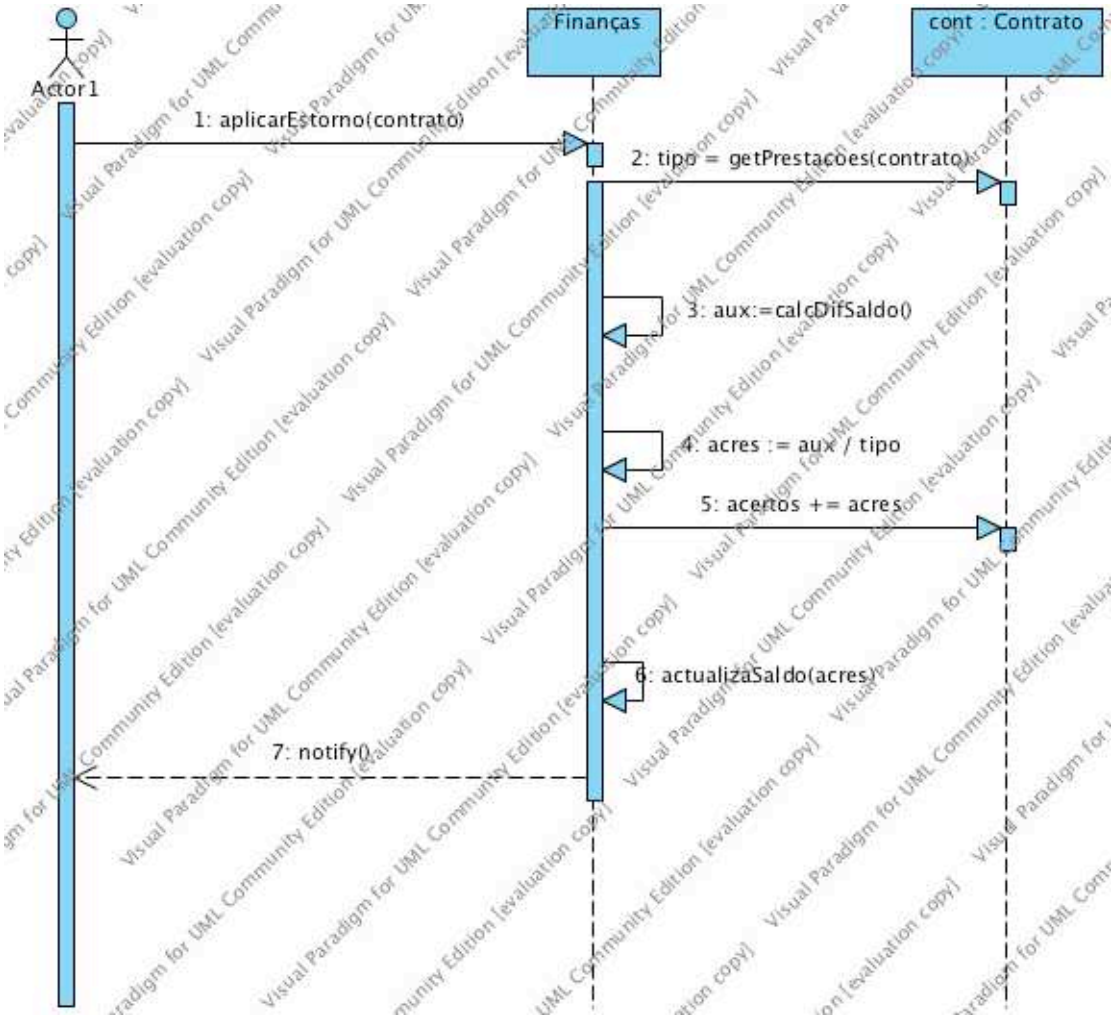


Diagrama de Sequência 4 : Aplicar Estorno

3.Diagrama de classes:

Os **diagramas de classes** são uma representação da estrutura de uma aplicação e relações das classes que servem de modelo para objectos.

É uma modelagem muito útil para o sistema pois define todas as classes que o sistema necessita possuir.

Conceitos

Classe: Elemento abstracto que representa um conjunto de objectos. A classe contém a especificação do objecto; suas características: atributos e métodos (acções / comportamentos).

- **Atributo:** Define características da classe tais como:
  - **Visibilidade;**
  - **Nome;**
  - **Tipo de dados;**
  - **Multiplicidade;**

- **Operação**: Função requerida a um objecto.

- **Associação**: Relacionamentos entre classes.

- **Navegação:** De onde vem as informações da classe e para onde vai.

Tipos de relacionamentos entre Classes:

Agregação

Demonstra que as informações de um objecto precisam ser complementadas de outra classe. Esta associação, representa uma relação forte entre as classes, isto

é, se a classe que "contém" for destruída não significa que a classe "contida" será.

É representada por uma linha com um diamante “branco” do lado da classe que contém.

Composição

Um tipo de agregação, onde um objecto pertence a outro.

Se a classe que "é dona" for destruída significa que a classe "contida" não fará sentido existir.

É representada por uma linha com um diamante cheio do lado da classe que contém.

Generalização

Também conhecida como herança, representa as dependências e hierarquias.

Em seguida faremos uma breve explicação do nosso Diagrama de Classes.

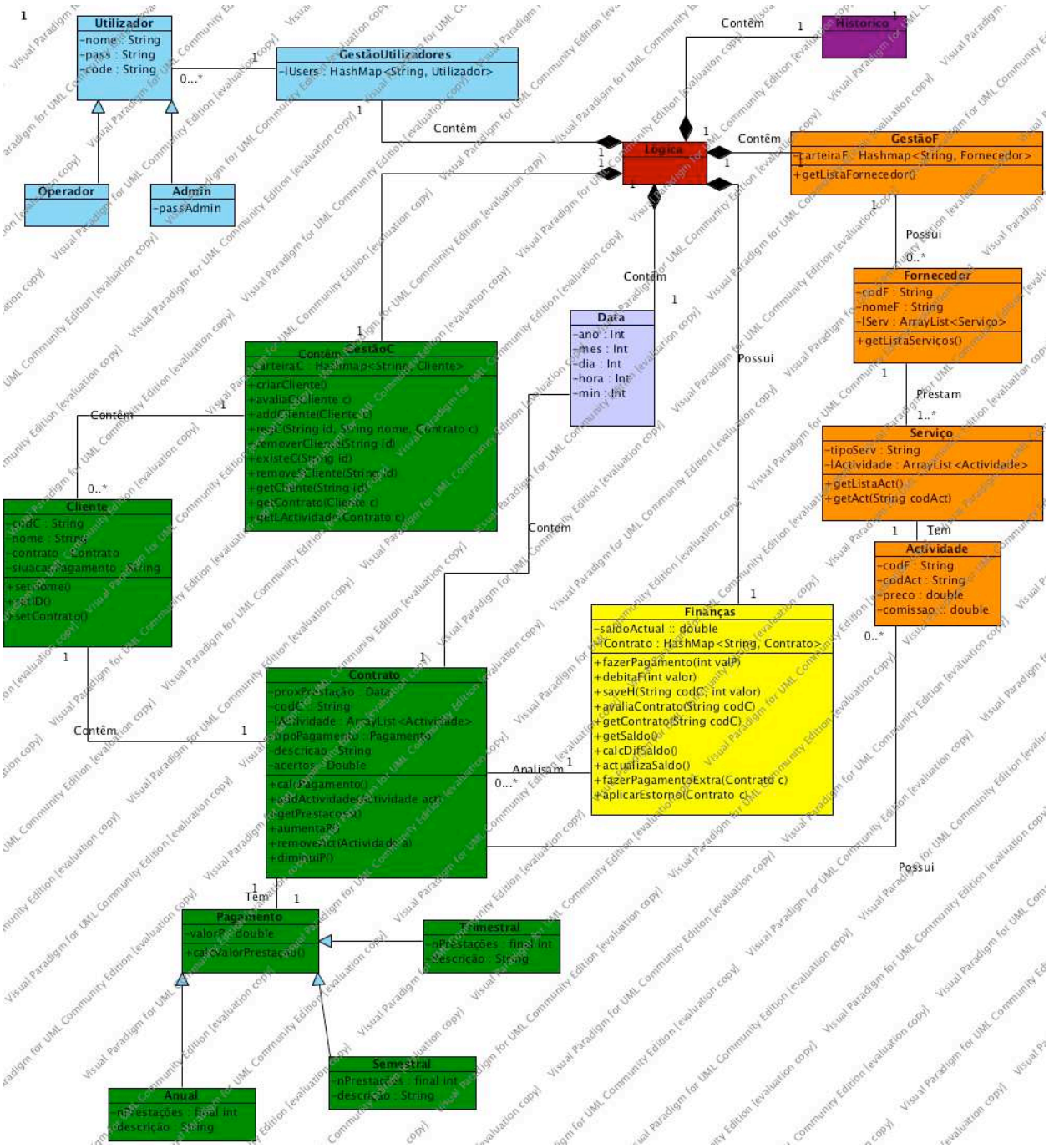


Figure 1 : Diagrama de Classes

Através do Diagrama de Classes acima representado é facilmente compreensível a arquitectura da GereComSaber. De um modo geral, o nosso sistema é subdividido em três subsistemas principais que possuem responsabilidades distintas:

• **Finanças** : Subsistema responsável por todas as funcionalidades que envolvam fundos monetários (pagamentos a fornecedores, acertos, estornos etc.) .

• **GestãoC :** Subsistema responsável por todas as acções referentes á Gestão dos Clientes da GereComSaber.

• **GestãoF :** Subsistema responsável por todas as actividades que digam respeito a Fornecedores da GereComSaber , actividades estas descritas na fase anterior do projecto nos diagramas de Use Case da Gestão de Fornecedores.

Todos os subsistemas referidos anteriormente possuem a sua própria API, o que garante o funcionamento individual e independente de cada um deles.

4.Decisões

Visto que para a realização e concretização com sucesso deste projecto, passamos a apresentar as várias decisões de implementação pelas quais estamos nos estamos a seguir e consequentemente toda a nossa modelação está de acordo com elas:

a) Finanças :

• Cada Cliente tem associado um valor da sua prestação global, o que significa que o valor da sua prestação é calculada englobando a totalidade das Actividades pertencentes ao seu contrato.

• A GereComSaber desconhece o estado financeiro de cada Fornecedor , pelo que um pagamento significa um aumento e redução nos fundos da GereComSaber. Esta acção é imediata. Os valores discriminados dos pagamentos são visíveis no histórico de operações.

• Dadas as decisões acima mencionadas o lucro da GereComSaber é determinado simplesmente pela consulta do saldo uma vez que os pagamentos são automáticos e o restante é o valor ganho pelas comissões pagas pelos fornecedores.

• Em caso de acertos, caso se trate de um Acréscimo é simplesmente actualizado o valor da prestação global do Cliente, neste caso obviamente irá aumentar. No caso de Estorno, é igualmente actualizado o valor da prestação, diminuindo, e o valor do campo Acerto do Cliente é actualizado com o respectivo Estorno.

b) Fornecedores :

• Relativamente ás comissões pagas pelos Fornecedores á GereComSaber, estas são individualizadas por Actividade, ou seja, cada Actividade consoante as suas características pode ter um valor de comissão diferente mesmo pertencendo ao mesmo tipo de Serviço.

• Todos os preços dos Serviços disponibilizados pelos fornecedores são apresentados aos Clientes inflacionados pela comissão que cada um deles designa individualmente para cada Actividade providenciada.

5. Conclusões e trabalho futuro

Nesta fase, foi possível apercebermo-nos de alguns erros de captura de requisitos da fase anterior, que resultaram num refinamento dos Use Case e no seu detalhe, mais precisamente na parte de Gestão de Clientes. A partir desta fase, é-nos possível com alguma facilidade, passar-mos a parte de implementação da aplicação, visto já ter-mos todas as classes, métodos e variáveis devidamente estudadas e documentadas através do diagrama de Classes.

7. Anexos

Em anexo acrescentamos todos os diagramas de sequencia do nosso sistema:

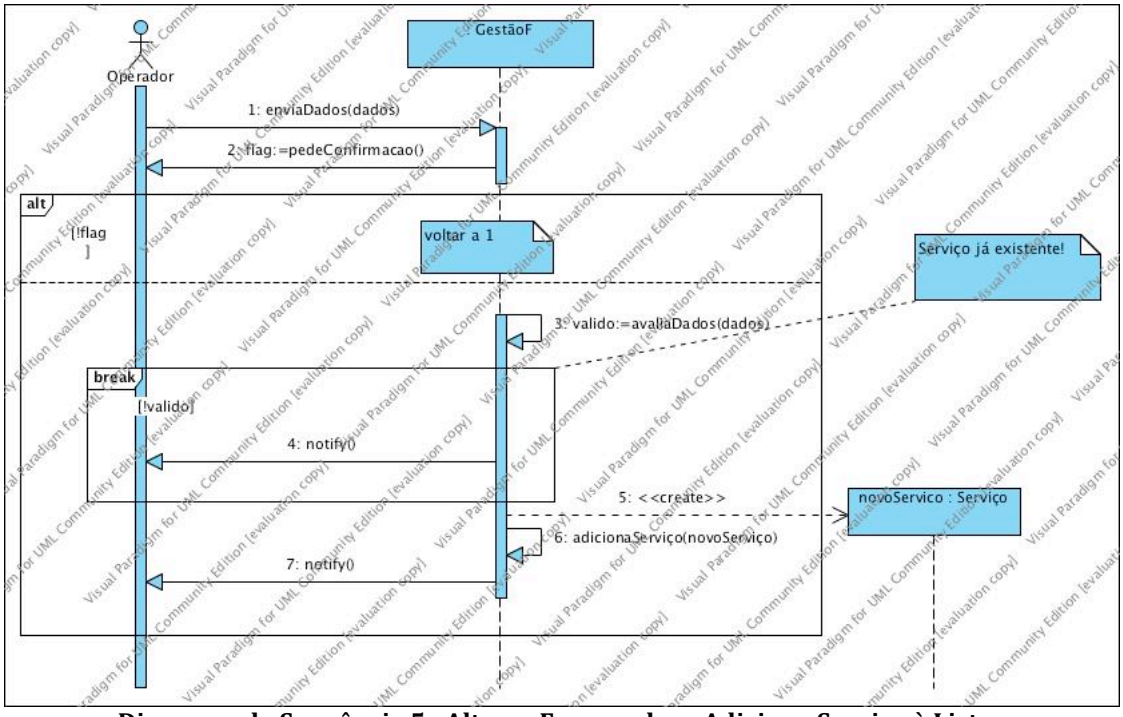


Diagrama de Sequência 5 : Alterar Fornecedor - Adiciona Serviço à Lista

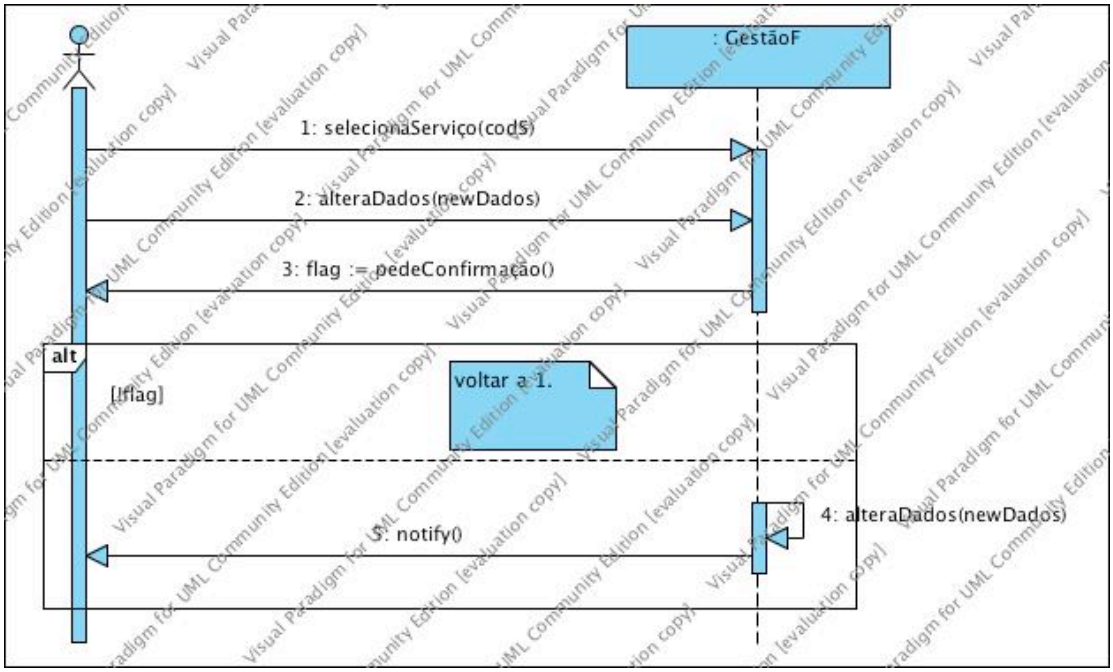


Diagrama de Sequência 6 : Alterar Fornecedor - Alterar Serviço à Lista

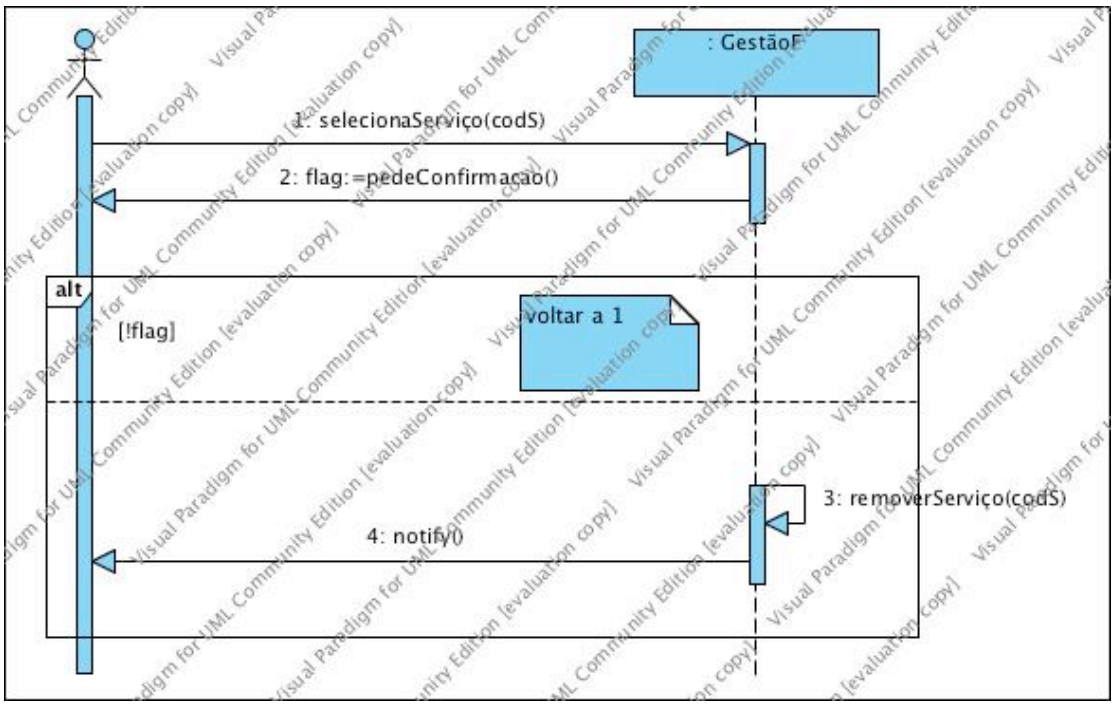


Diagrama de Sequência 7 : Alterar Fornecedor - Remove Serviço à Lista

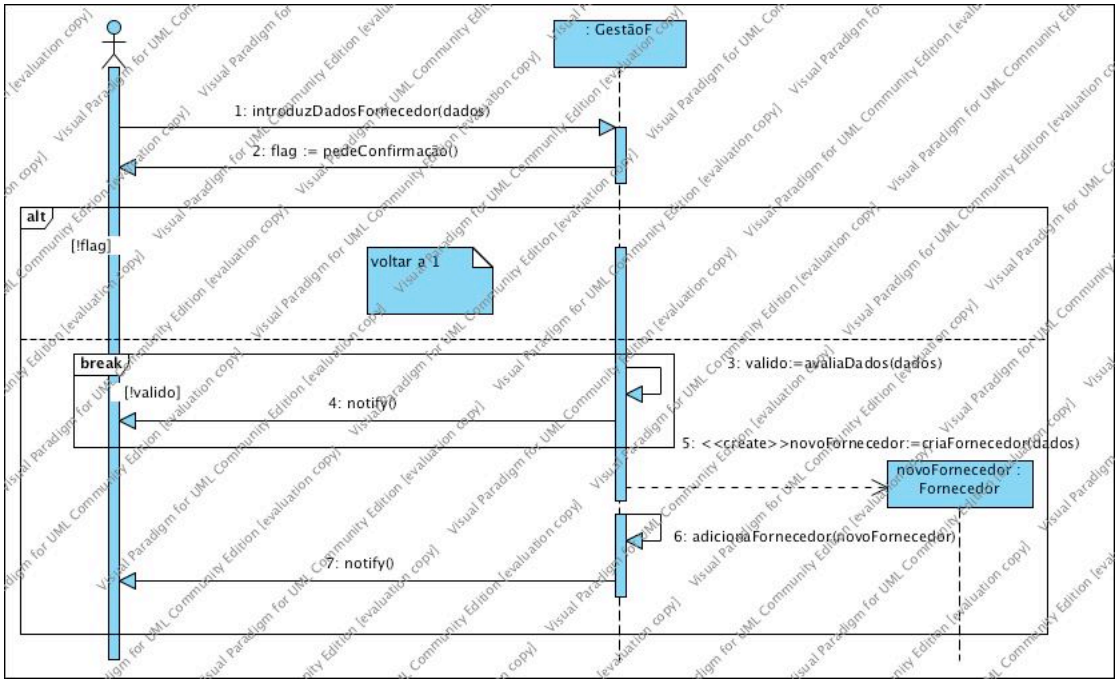


Diagrama de Sequência 8 : Gerir Fornecedores - Criar Fornecedor

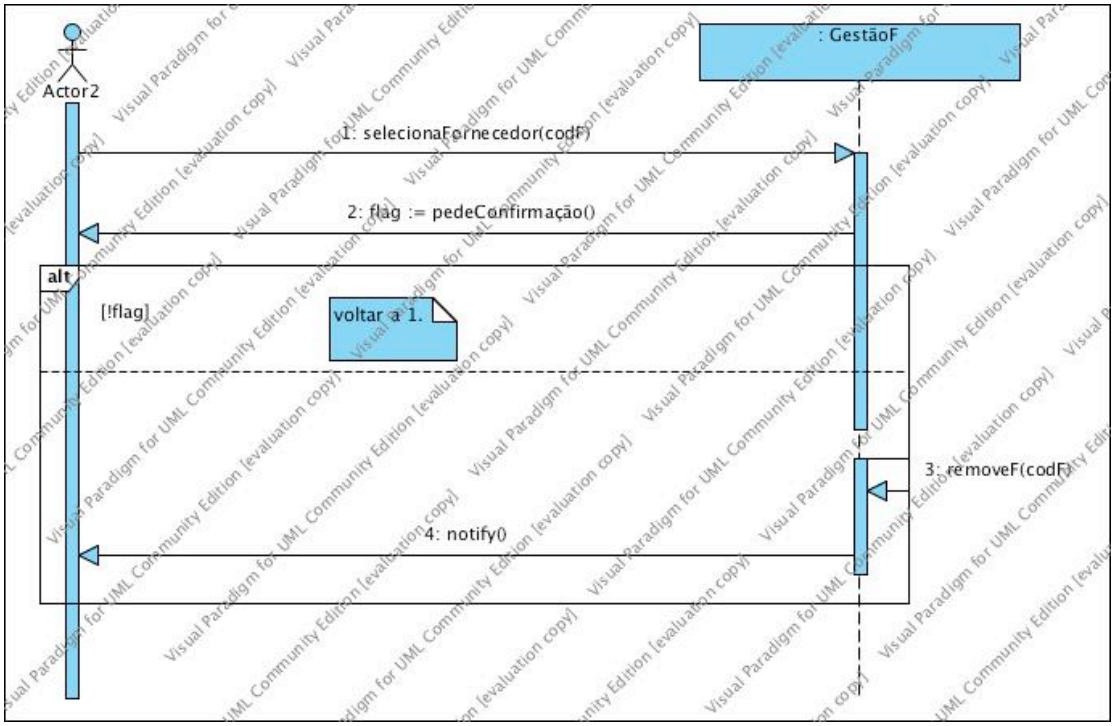


Diagrama de Sequência 9 : Gerir Fornecedores - Remover Fornecedor

Diagrama de Sequência 6 : Gerir Pagamentos - Calcular margem de lucro

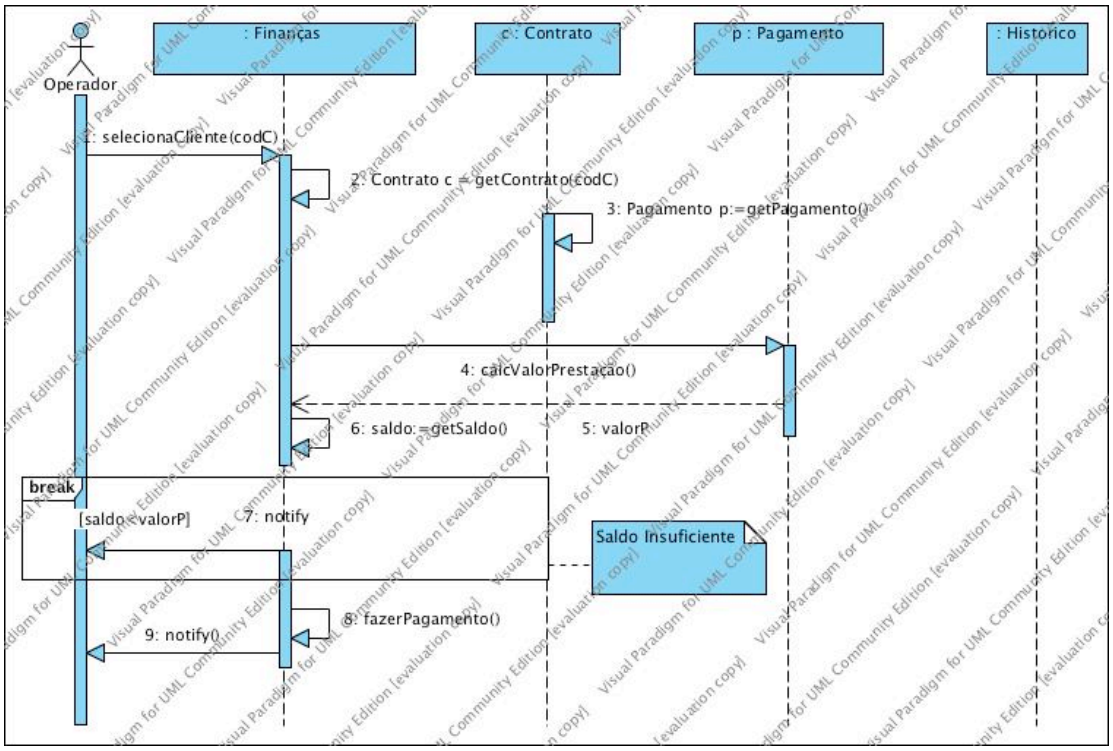


Diagrama de Sequência 10 : Gerir Pagamentos - Calcular margem de lucro

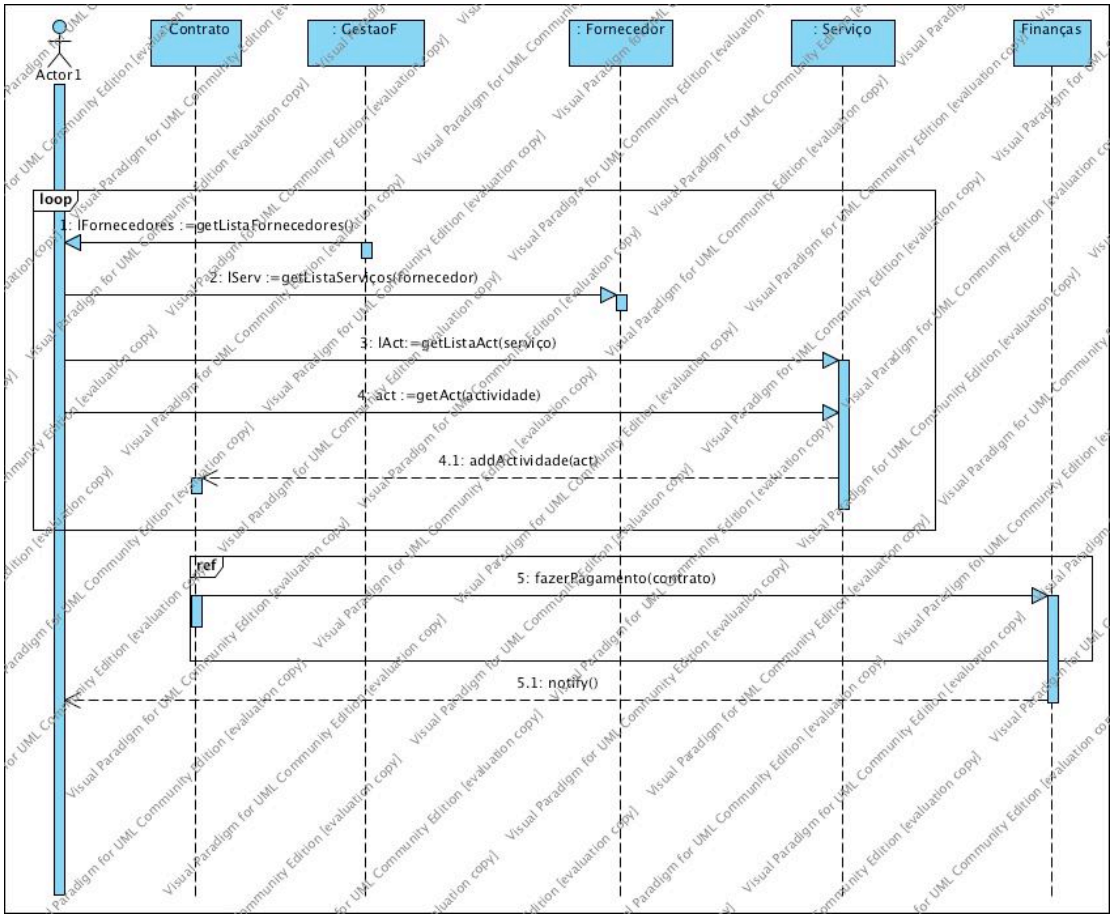


Diagrama de Sequência 11 : Gerir Clientes - Adicionar novo Cliente

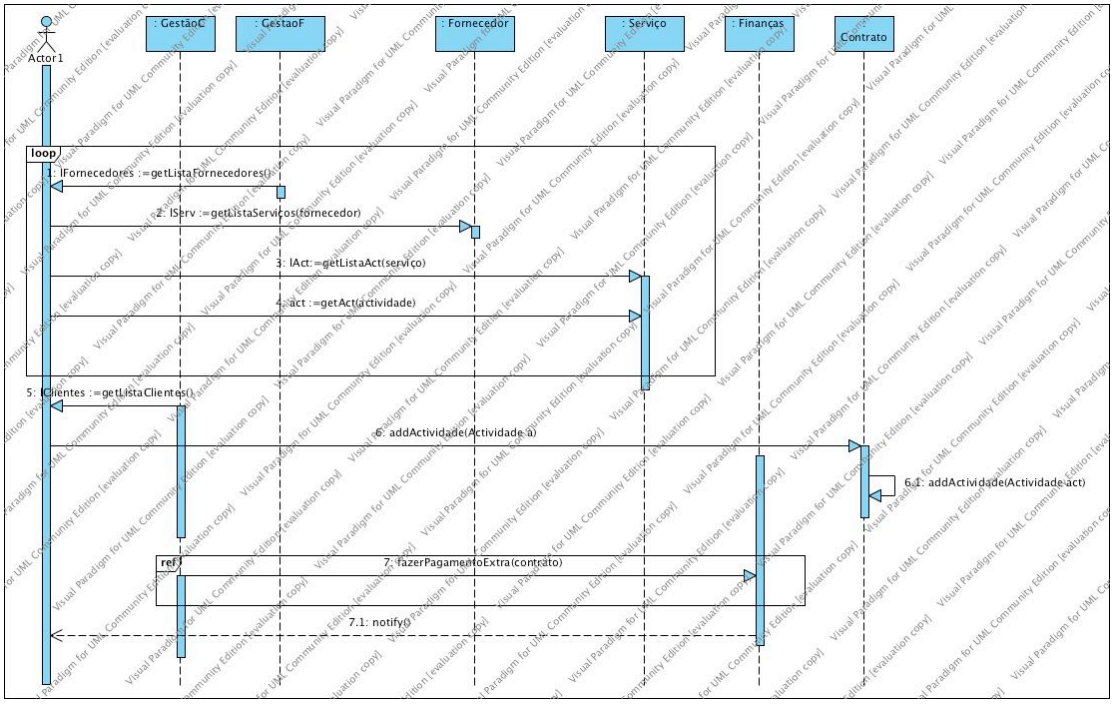


Diagrama de Sequência 12 : Gerir Clientes - Adicionar Serviço Cliente

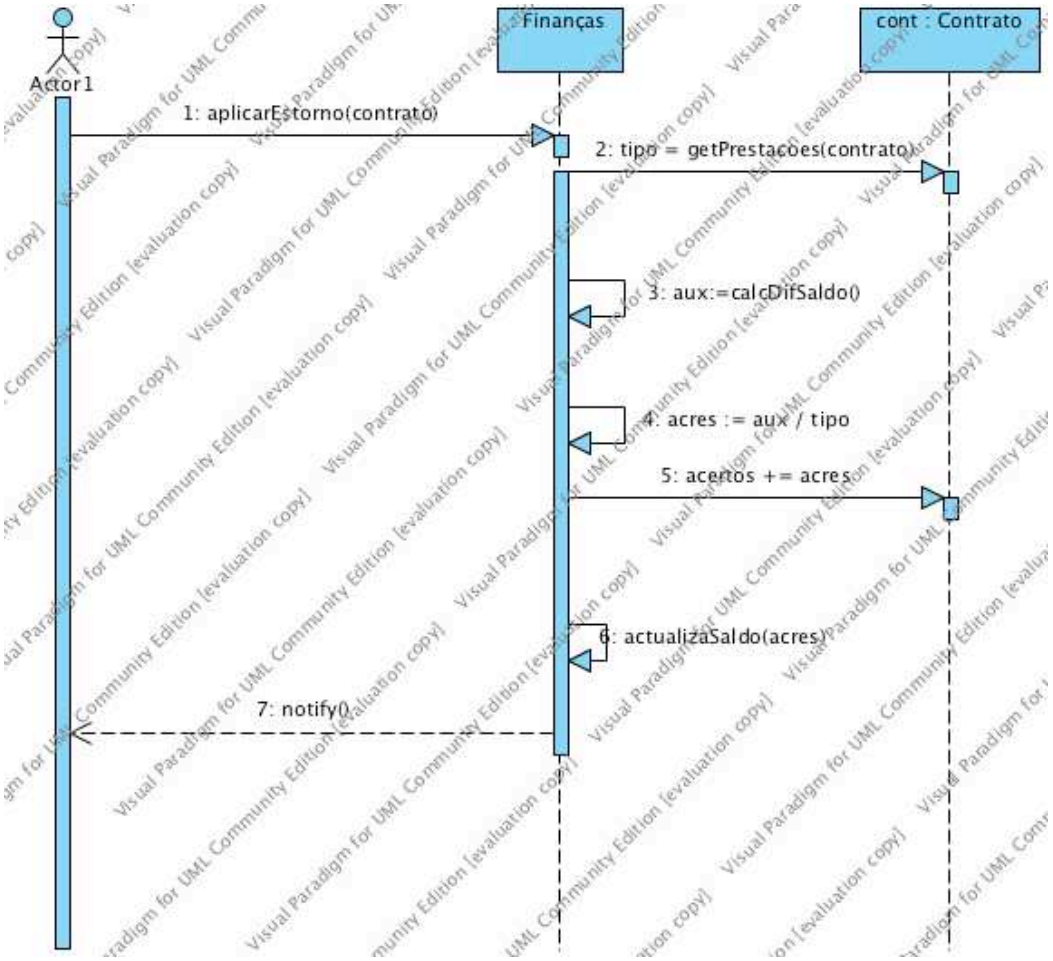


Diagrama de Sequência 13 : Gerir Clientes - Aplicar Estorno

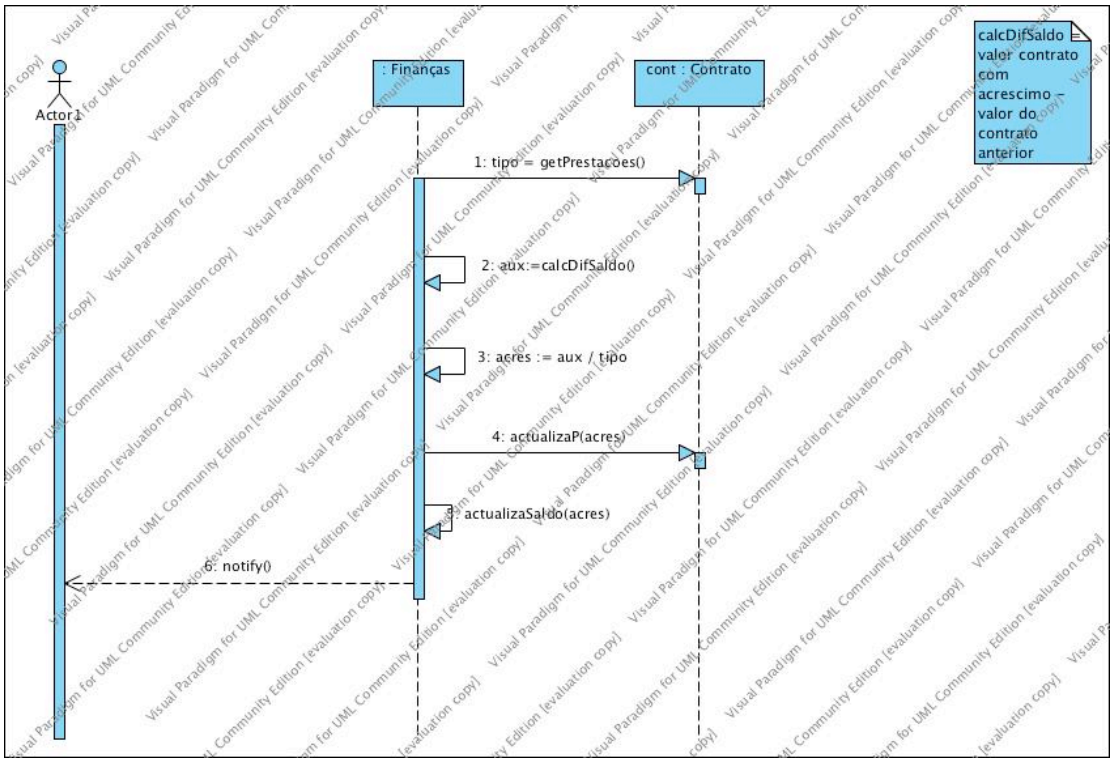


Diagrama de Sequência 14 : Gerir Clientes - Aplicar Pagamento Extra

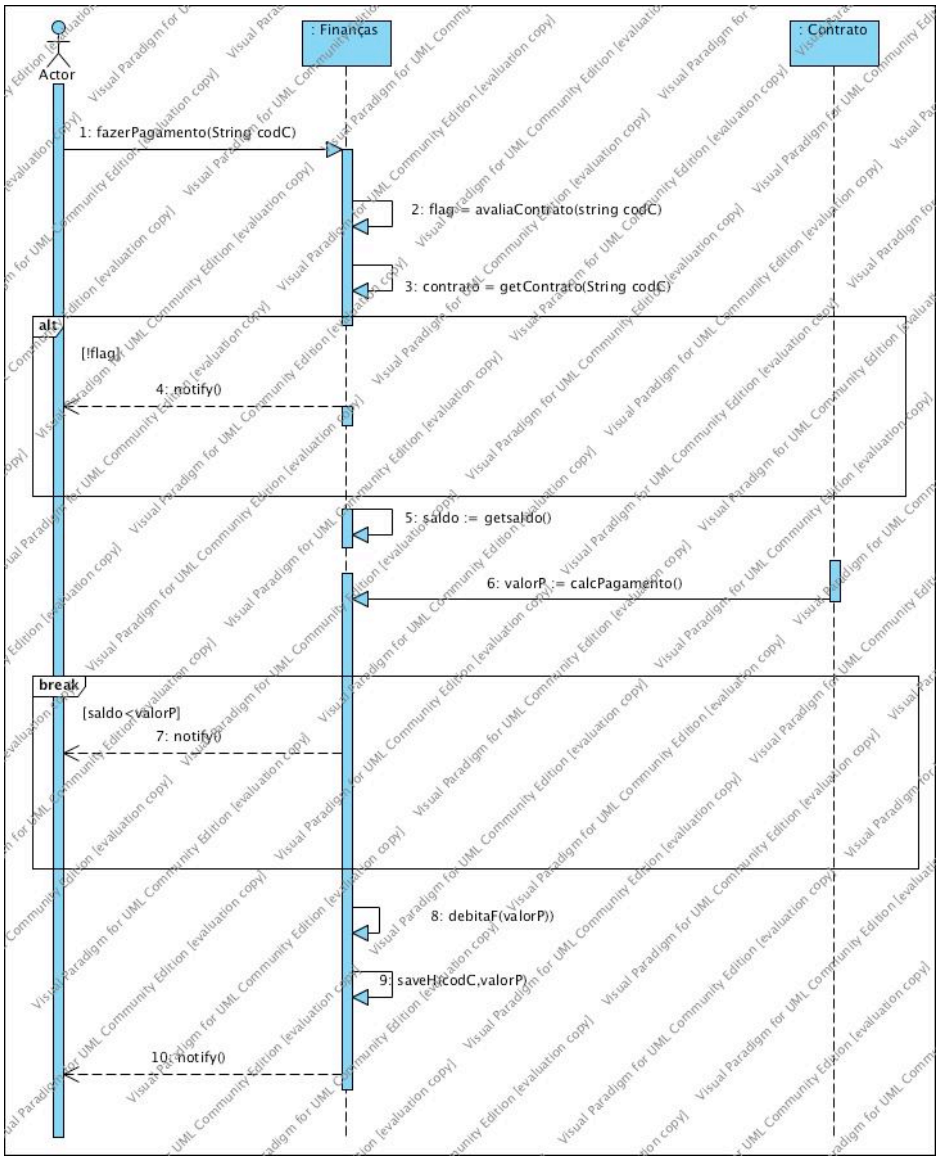


Diagrama de Sequência 15 : Gerir Clientes - Fazer Pagamento

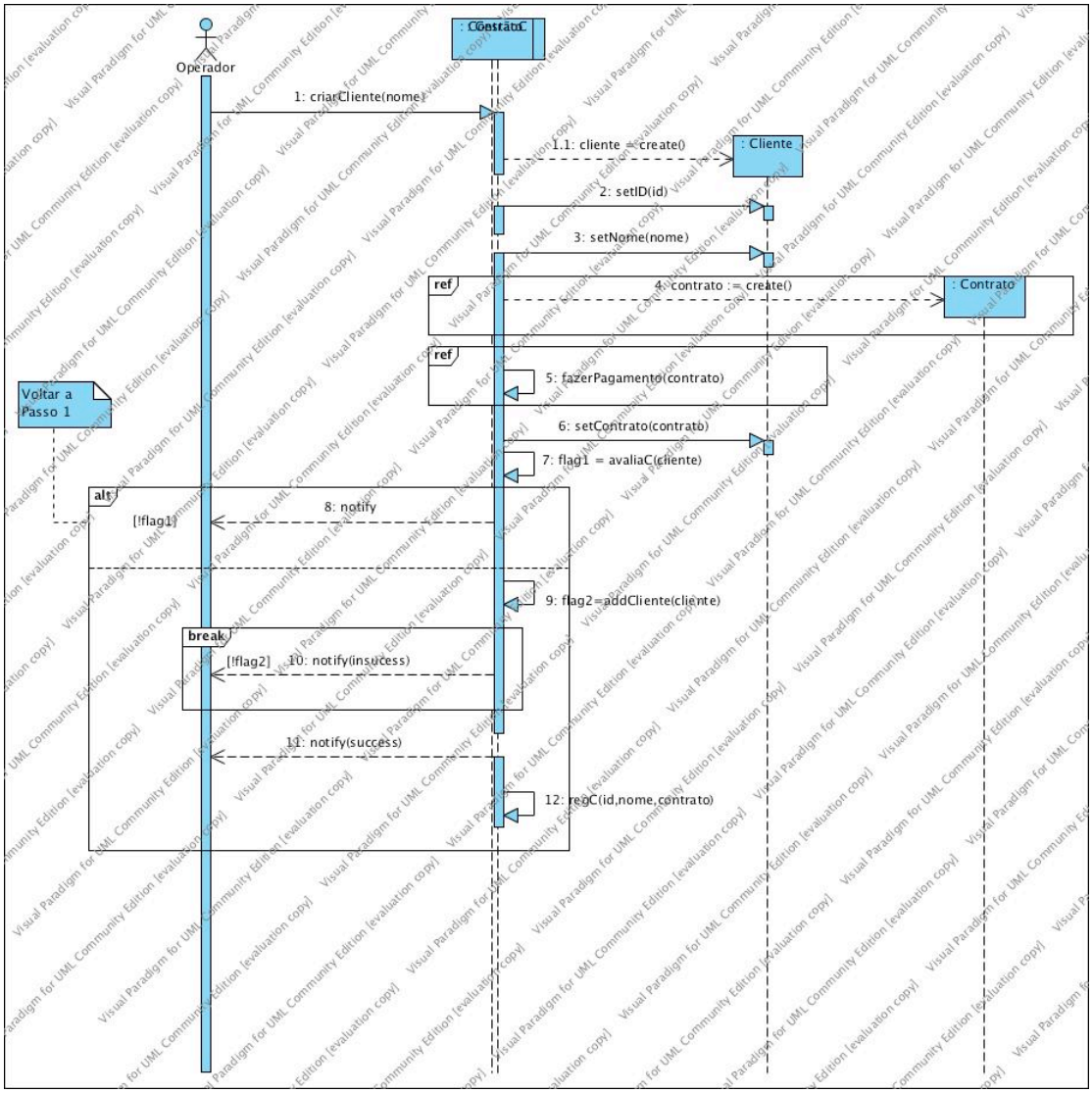


Diagrama de Sequência 16 : Gerir Clientes - Criar Cliente

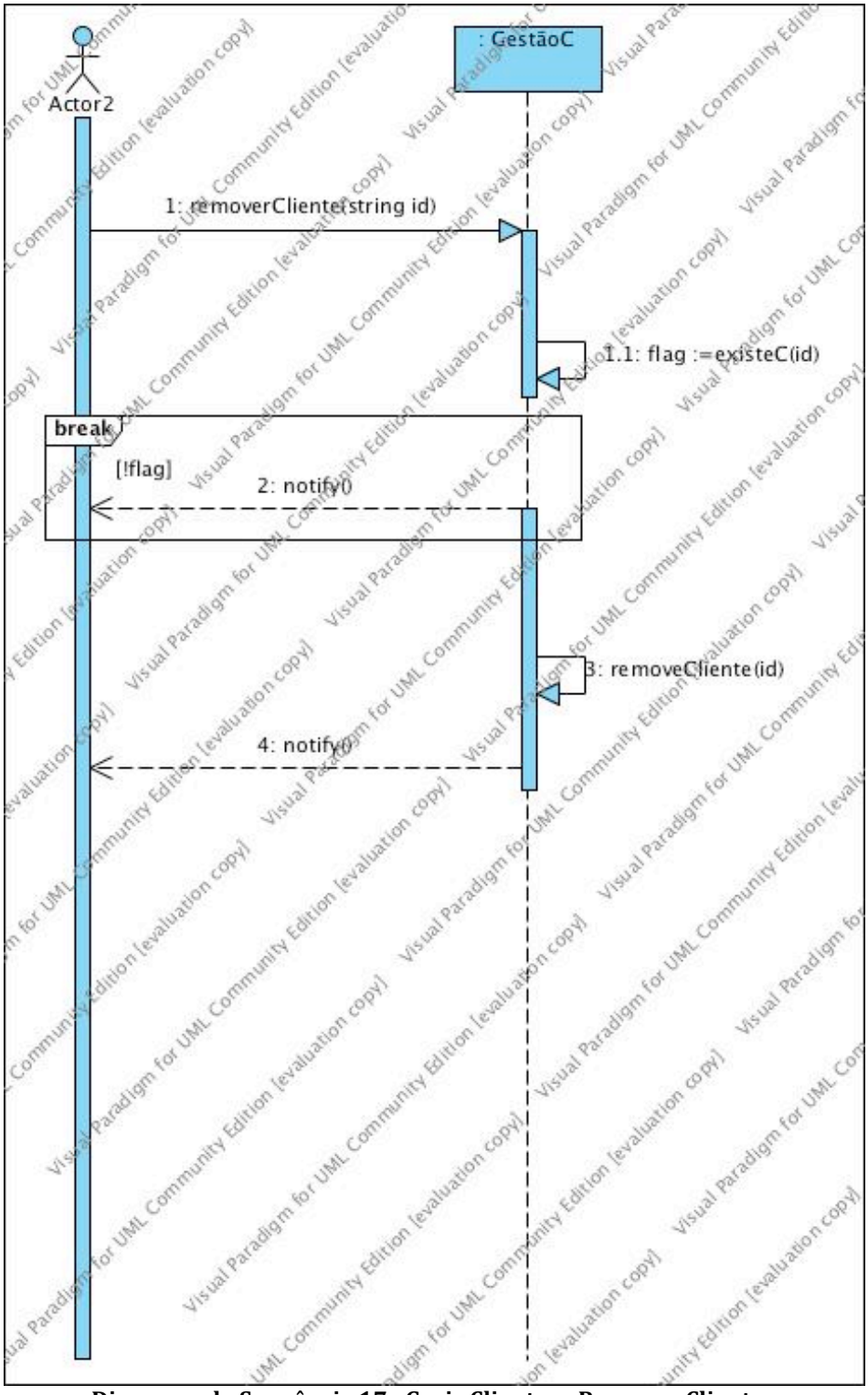


Diagrama de Sequência 17 : Gerir Clientes - Remover Cliente

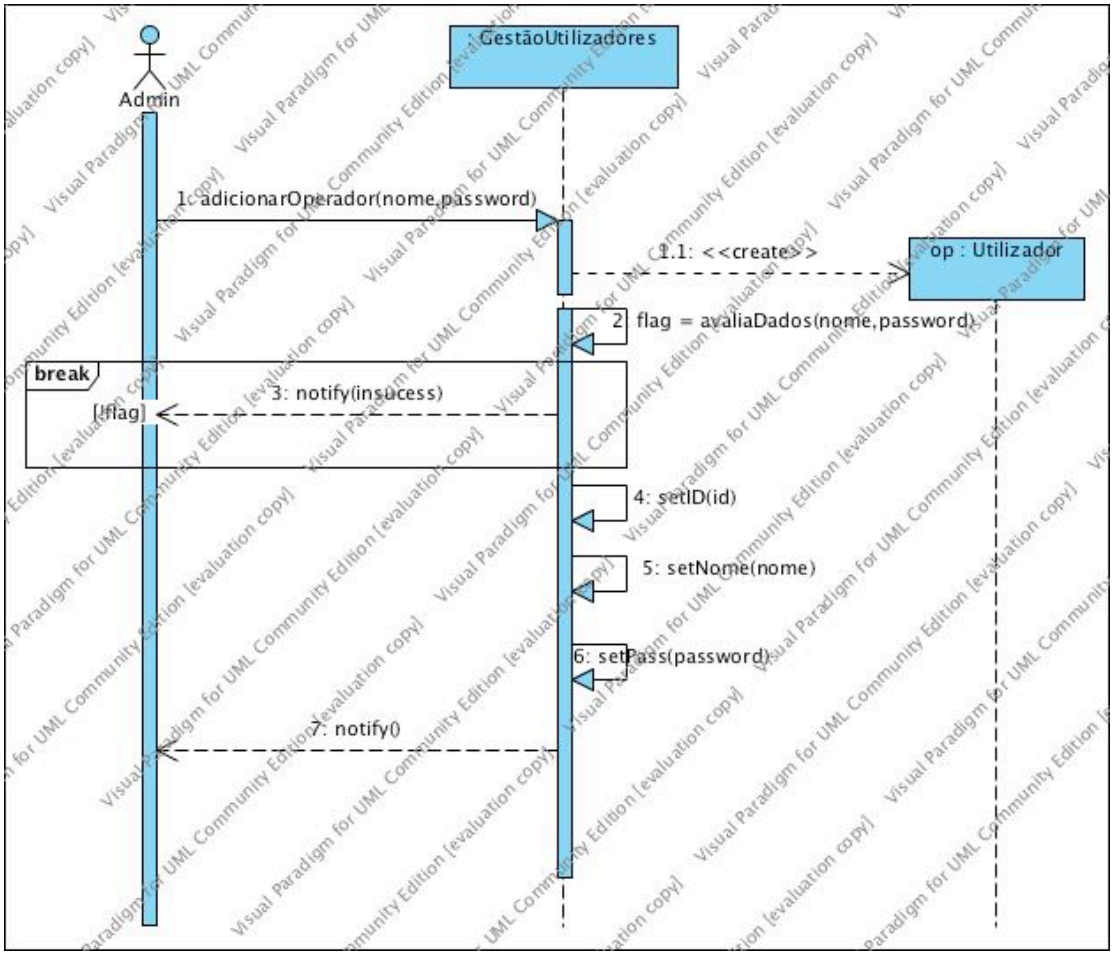


Diagrama de Sequência 18 : Gerir Operadores - Adicionar Operador

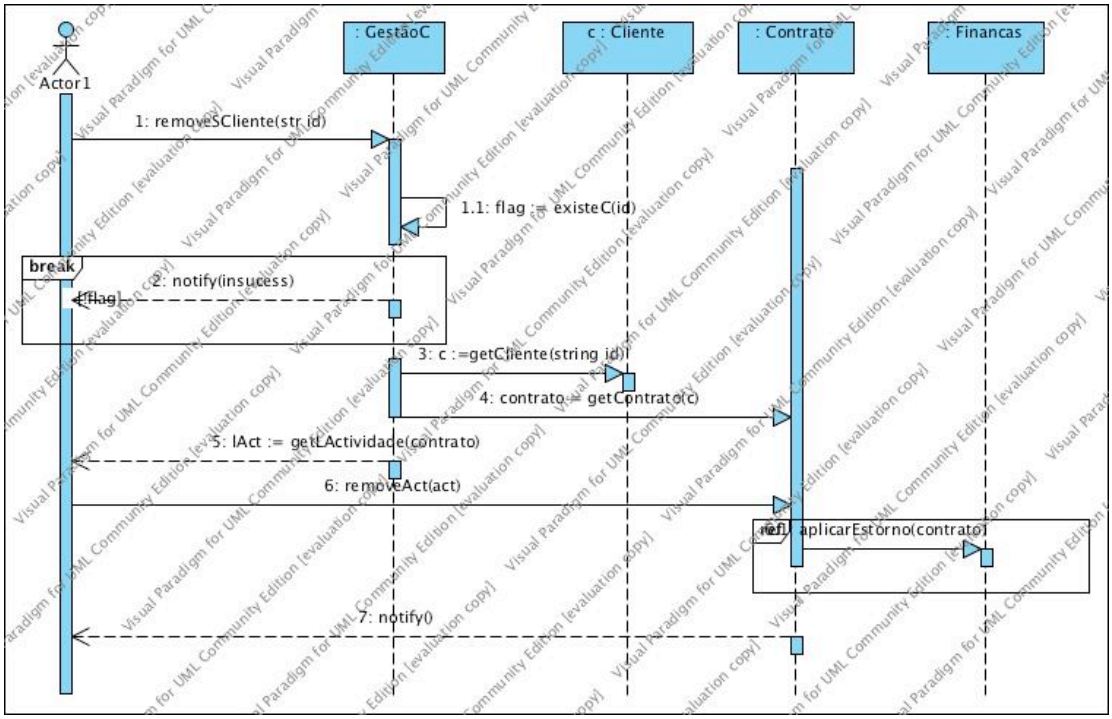


Diagrama de Sequência 19 : Gerir Clientes - Remover Serviço Cliente