



Universidade do Minho

Escola de Engenharia

Mestrado Integrado em Engenharia Informática

Unidade Curricular de Laboratórios de Informática IV

Ano Lectivo de 2017/2018

**JJMS - Serviço de logística com
entrega personalizada e
funcionamento 24/7**

João Pedro Ferreira Vieira (a78468), José Carlos Lima Martins (a78821),
Miguel Miranda Quaresma (a77049), Simão Paulo Leal Barbosa (a77689)

Junho, 2018

LI4

1718

Data de Recepção	
Responsável	
Avaliação	
Observações	

JJMS - Serviço de logística com entrega personalizada e funcionamento 24/7

**João Pedro Ferreira Vieira (a78468), José Carlos Lima Martins (a78821),
Miguel Miranda Quaresma (a77049), Simão Paulo Leal Barbosa (a77689)**

Resumo

O presente documento apresenta a fundamentação e o processo de desenvolvimento de um sistema de software usado numa empresa de logística (**JJMS**).

Numa primeira instância é contextualizado o problema no qual o sistema de software será desenvolvido, neste caso uma empresa de logística com serviço personalizado de entrega 24/7. De seguida apresenta-se uma análise à sua viabilidade bem como uma justificação para o seu desenvolvimento, especificando ainda os serviços que farão parte deste sistema e a interatuação entre os mesmos. São ainda identificados os recursos necessários ao desenvolvimento e manutenção deste sistema.

No fim desta fase inicial, são definidas as medidas de sucesso para o mesmo. Por forma a agilizar o processo de desenvolvimento é ainda apresentado um plano de desenvolvimento com recurso a diagramas Gantt.

No que diz respeito à fase de especificação, são apresentados os use cases extraídos do problema a resolver, especificando a fundo cada um deles por forma a facilitar a implementação do projeto posteriormente. Desta forma são utilizadas tabelas de especificação de use cases, diagramas de sequência, prototipagem de interfaces e respectivas máquinas de estado para demonstrar as trocas de estado entre as diferentes interfaces, diagramas de classes e por fim modelação conceptual e lógica para a BD do sistema.

Por fim, na fase de implementação, foi desenvolvida a página Web bem como o modelo físico da BD de modo a suportar a página, sendo explicado os passos realizados bem como realizadas conclusões acerca do trabalho realizado.

Área de Aplicação: Fundamentação, Especificação, Implementação

Palavras-Chave: JJMS, Diagrama de Gantt, Aplicação Mobile, Página Web, Sistema de Software, Plano de desenvolvimento, Diagrama de Sequência, Diagrama de Classes, Prototipagem, Máquina de Estados, DAO, Use Case, Modelo ER, Modelo Lógico da BD, C#, MVC, ASP.NET, SQL Server, EntityFramework, Table-Per-Hierarchy

Índice

Resumo	3
Índice	4
Índice de Figuras	6
Índice de Tabelas	8
Introdução	9
Contextualização	10
Motivação e Objectivo	10
Definição da identidade do sistema a desenvolver	11
Justificação, viabilidade e utilidade do sistema	11
Identificação dos recursos necessários	12
Modelo do sistema a implementar	13
Definição de medidas de sucesso	14
Plano de desenvolvimento	14
Especificação	14
Implementação	15
Especificação	15
Requisitos	16
Utilizador	16
Funcionais	16
Não Funcionais	17
Sistema	17
Funcionais	17
Não Funcionais	18
Use Cases	19
Especificação dos Use Cases:	20
Prototipagem dos Use Cases	21
Diagramas de Sequência	22
Diagramas de Sequência dos Use Cases (Uses Cases -> Diagramas de Sequência)	22

Diagramas de Sequência de Sistema dos Use Cases (Definição de funções)	24
Máquinas de Estado da Prototipagem	25
Diagramas de Sequência das funções definidas	26
Diagrama de Classes	34
Modelo (de Base de Dados) Conceptual	36
Modelo (de Base de Dados) Lógico	42
Implementação	43
Arquitetura MVC e Organização	43
Models	43
Data	44
Controllers	45
Views	46
EntityFramework usada com Code First Approach	49
Adaptações à especificação	53
Diagrama de Classes sem persistência	54
Diagrama de Classes com persistência	56
Funcionalidades efetivamente implementadas vs medidas de sucesso	57
Instalação	58
Conclusões e Trabalho Futuro	59
Referências Bibliográficas	60
Lista de Siglas e Acrónimos	62
Anexos	63
Especificação de Use Cases	63
Prototipagem dos Use Cases	71
Diagramas de Sequência dos Use Cases (Uses Cases -> Diagramas de Sequência)	77
Diagramas de Sequência dos Use Cases (Definição de funções)	86
Máquinas de Estado da Prototipagem	96
Diagramas de Sequência das funções definidas	102

Índice de Figuras

Figura 1 - Logo da Empresa	10
Figura 2 - Maquete	12
Figura 3 - Diagrama Gantt da especificação	14
Figura 4 - Diagrama Gantt da implementação	14
Figura 5 - Diagrama de Use Cases	19
Figura 6 - Prototipagem de “Requisitar Encomenda”	21
Figura 7 - Diagrama de Sequência de “Requisitar Encomenda”	23
Figura 8 - Diagrama de Sequência de “Delegar Funcionários”	23
Figura 9 - Diagrama de Sequência de “Requisitar Encomenda” com API	24
Figura 10 - Diagrama de Sequência de “Delegar Funcionários” com API	25
Figura 11 - Máquina de Estados de “Requisitar Encomenda”	26
Figura 12 - Diagrama de Sequência de “estaBloqueado”	26
Figura 13 - Diagrama de Sequência de “cartaoValido”	27
Figura 14 - Diagrama de Sequência de “idForn”	28
Figura 15 - Diagrama de Sequência de “setEncomenda”	28
Figura 16 - Diagrama de Sequência de “newEncomenda”	29
Figura 17 - Diagrama de Sequência de “newCartaoCredito”	29
Figura 18 - Diagrama de Sequência de “luhn_check”	30
Figura 19 - Diagrama de Sequência de “delegarFuncionario”	31
Figura 20 - Diagrama de Sequência de “getDestinoEnc”	31
Figura 21 - Diagrama de Sequência de “enviarEmail”	32
Figura 22 - Diagrama de Sequência de “getMoradaForn”	32
Figura 23 - Diagrama de Sequência de “getEstadoEncomendal”	33
Figura 24 - Diagrama de Sequência de “getEstadoEncomendaS”	33
Figura 25 - Diagrama de Sequência de “getUserEmail”	34
Figura 26 - Diagrama de Sequência de “getIdForn”	34
Figura 27 - Diagrama de Classes	35
Figura 28 - Diagrama de Classes com persistência	35
Figura 29 - Modelo ER da BD	41

Figura 30 - Modelo Lógico da BD	42
Figura 31 -Interface da escolha e agendamento da Encomenda	46
Figura 32 - Interface da introdução dos dados de pagamento	47
Figura 33 - Interface para alterar os dados	48
Figura 34 - Interface para consultar histórico	49
Figura 35 - Modelo Lógico com EntityFramework e Code First	50
Figura 36 - Tabelas antes	52
Figura 37 - Tabelas após	52
Figura 38 - Novo Diagrama de Classes sem persistência	54
Figura 39 - Novo Diagrama de Classes com persistência	56

Índice de Tabelas

Tabela 1 - Especificação do use case “Requisitar Encomenda”	20
Tabela 2 - Caracterização de Entidades	36
Tabela 3 - Caracterização de Relacionamentos	37
Tabela 4 - Caracterização de Atributos	38
Tabela 5 - Caracterização de chaves candidatas e primárias	40

1. Introdução

Este documento tem como objetivo mostrar os métodos e ideias por detrás do desenvolvimento do projeto em questão, explicitando todos os modelos construídos pelo grupo e considerados como importantes para o sistema a implementar. Um dos principais objetivos da segunda fase de trabalho passa por facilitar e dar uma maior ideia da aplicação pretendida pela JJMS ao programador ou equipa de programação que vão implementar o sistema na fase seguinte, estando assim mais perto de perceber se a aplicação/sistema por estes implementada vai de encontro ao pretendido da parte do cliente.

Com todo este processo de especificação do que é pretendido desenvolver, o tempo de programação é reduzido substancialmente, visto que, grande parte da criação de soluções para problemas foi dispensado nesta fase de um modo sustentado e suportado em diagramas, algo por vezes mais simples de perceber tanto para o programador bem como para o cliente do que o código propriamente dito mesmo que bem documentado.

De seguida será apresentada a primeira fase deste projeto sendo depois, apresentado o módulo sobre a especificação deste mesmo projeto e por fim a fase de implementação.

1.1. Contextualização

A JJMS é uma empresa de logística sediada em Braga e alargada de forma contínua e crescente a todo o Portugal Continental e ilhas. O nome da empresa (JJMS) advém das iniciais dos seus fundadores. A JJMS trabalha diretamente com determinados fornecedores no transporte de encomendas relacionadas com compras online, preocupando-se apenas com o transporte das encomendas entre o fornecedor e o cliente. A empresa possui uma grande frota de transporte de encomendas, e um vasto leque de profissionais, tendo a sua central de distribuição estrategicamente posicionada em Lisboa.(tendo em conta o aeroporto).

A JJMS distingue-se de todas as outras empresas devido ao seu serviço personalizado, permitindo ao cliente escolher a hora a que pretende receber ou levantar determinada encomenda, com a componente inovadora de que qualquer hora pode mesmo ser especificada, tendo em conta o funcionamento 24/7 da JJMS (disponível 24 horas por dia durante os 7 dias da semana).

A empresa foi pensada e implementada com este intuito devido à quantidade de empresas de logística que existem e que bem conhecemos, tendo mesmo algumas um grande poder no

mercado em que se inserem, surgindo assim a necessidade de incluir alguma característica que torne a JJMS uma empresa diferente de todas as outras e com algo inovador, de forma a não ser apenas mais uma empresa no meio de tantos “tubarões”.



Figura 1 - Logo da Empresa

1.2. Motivação e Objectivo

A JJMS tem vindo a apresentar problemas graves nos últimos tempos, levando a claras consequências quer na satisfação dos clientes quer nas receitas da empresa. Entre estes problemas estão:

- Entrega de encomendas em mau estado
- Atrasos significativos nas entregas das encomendas, o que até levou ao rompimento de certas relações com fornecedores
- Falta de canais de comunicação entre os clientes e a empresa
- Gestão ineficiente dos recursos (vários funcionários encarregues da mesma encomenda), derivado do facto de este ser efetuado por funcionários ao invés de um algoritmo apropriado para a distribuição das encomendas
- Sistema de requisição de encomendas pouco intuitivo/*user-friendly*

Como tal foi tomada a decisão de redesenhar o sistema usado recorrendo, para isso, a duas vertentes/interfaces mobile e web, com o objetivo de corrigir todos estes problemas.

1.3. Definição da identidade do sistema a desenvolver

O sistema a desenvolver apresentará duas vertentes de destaque. A primeira será desenvolvida na ótica dos funcionários, por forma a controlar o estado (**com o fornecedor > no centro de distribuição > em trânsito > entregue**) das encomendas. Para além disto, os funcionários deverão ter informação na aplicação referente à forma como alcançar o domicílio dos clientes e dos fornecedores.

A entrega das encomendas aos clientes e a recolha das mesmas nos fornecedores é delegada aos funcionários da empresa através de um algoritmo que tendo em conta a zona a que estes se encontram atribuídos, e o número de encomendas pelas quais são responsáveis, as distribui pelos funcionários de forma equitativa. Isto permite optimizar o uso dos recursos disponíveis (combustível, homem-hora) e ter uma gestão mais eficiente das encomendas a efetuar. Ao ser

delegada uma encomenda a um funcionário o mesmo recebe um email com os detalhes da mesma.

No que diz respeito ao lado dos clientes, após efetuarem compras num dos fornecedores podem requisitar e agendar a entrega das mesmas através da aplicação e do sistema Web, fazer o rastreio (*tracking*) das suas encomendas e realizar uma avaliação quanto à qualidade do serviço (tanto da parte do funcionário que entrega bem como do estado em que a encomenda foi entregue). Deve ser possível também ao cliente efetuar o seu registo, alterar os seus dados, bem como consultar as encomendas já realizadas e as faturas referentes às mesmas.

A interface com o utilizador, quer funcionário quer cliente, será feita mediante os seus dispositivos móveis e computadores.

1.4. Justificação, viabilidade e utilidade do sistema

As características que a JJMS apresenta justificam, por si só, o desenvolvimento deste sistema (tanto para clientes como para funcionários), dado o funcionamento 24/7 apresentado com o agendamento da altura da entrega das encomendas com os clientes.

O desenvolvimento tanto de uma *app mobile* como de uma página *web* com o funcionamento pretendido é viável porque o investimento inicial compensa a longo prazo, devido a uma maior satisfação dos clientes com os serviços prestados e uma melhor gestão das encomendas. Este sistema leva também a uma redução nos custos com funcionários, visto que com a inexistência do mesmo, alguém teria que ser responsável e tratar dos assuntos que este facilita de forma gratuita (como os funcionários que gerem os agendamentos das encomendas), excluindo no entanto o custo de implementação e manutenção (tal como energia, custos informáticos, entre outros) deste sistema. Por fim, devido ao uso de aparelhos eletrónicos, já pertencentes a funcionários bem como a clientes, tais como smartphones e portáteis, não há necessidade de um investimento nos mesmos, reduzindo a quantidade de equipamentos necessários adicionais ao funcionamento do sistema.

1.5. Identificação dos recursos necessários

Para o desenvolvimento deste sistema, identificamos como necessários os seguintes recursos:

- Equipa de **programadores** com conhecimento em desenvolvimento Web e Mobile
- **Cliente de chat** para comunicação entre membros da equipa de desenvolvimento (ex: Slack, Discord, Telegram, etc)
- Equipa de **QA** para testar o software desenvolvido
- **Ferramentas** de desenvolvimento **multi-plataforma** derivado do uso de sistemas Windows e Linux para o desenvolvimento do projeto

- **Investimento** que cubra os custos associados ao funcionamento da equipa de desenvolvimento
- Microsoft Office para desenvolvimento de Diagramas Gantt, relatórios e documentação
- Microsoft Project de modo a delegar as tarefas entre os membros
- Microsoft SQL Server como sistema de gestão da base de dados
- **Reuniões semanais** dos membros da equipa
- **Intermediário** entre a JJMS e a equipa de desenvolvimento (Sr. Carvalho) que permita obter uma visão mais fiel do sistema a desenvolver bem como, **reuniões** esporádicas com **trabalhadores da empresa**
- Realização de **inquéritos ao público** (alvo) por forma a identificar os pontos de maior importância no sistema referido

1.6. Modelo do sistema a implementar

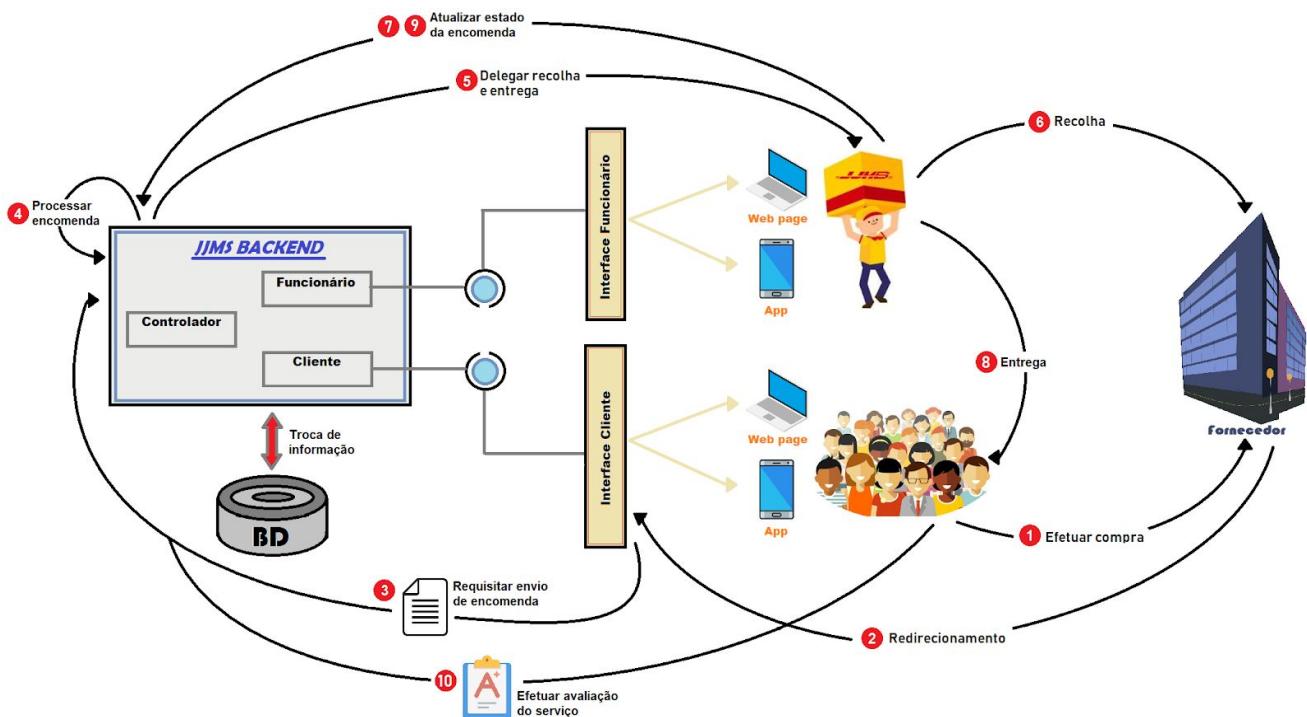


Figura 2 - Maquete

O sistema implementado tem como utilizador principal os clientes que adquirem produtos dos fornecedores afiliados com a JJMS. Quando esta aquisição, e após o pagamento do produto, estes são redirecionados para a página web ou para a aplicação mobile, caso se encontrem a usar o seu dispositivo móvel. De seguida preenchem um formulário com todos os dados necessários para o envio da encomenda. Após isso, este formulário é processado no sistema e, usando o algoritmo de distribuição, é delegado a um funcionário a recolha da encomenda no fornecedor e entrega da mesma no centro de distribuição, e através da mesma ideia de

delegação, eleger um funcionário que realize a entrega da encomenda desde a central ao cliente. Esta delegação é notificada ao funcionário através de um email com os dados da encomenda. Os funcionários responsáveis pelo movimento da encomenda vão atualizando o seu (da encomenda) estado. Após a entrega da encomenda, o funcionário que ficou encarregue da entrega atualiza o estado uma última vez para ‘Entregue’, sendo debitado o valor correspondente ao cliente. Por fim, o cliente avalia o serviço prestado, tanto o estado da encomenda bem como a prestação do serviço pelo funcionário.

1.7. Definição de medidas de sucesso

Como medidas de sucesso para o sistema de software em questão destacamos as seguintes:

- conclusão do projeto dentro do prazo estabelecido (28 de maio)
- custo de desenvolvimento do projeto em termos de homem-horas dentro do esperado
- garantia de um serviço fiável com tolerância de entrega de X minutos
- medidas de segurança no serviço web e na aplicação que respeitem os standards mínimos implementados na indústria (encriptação dos dados do utilizador e em trânsito, password hashing, etc)

1.8. Plano de desenvolvimento

O plano de desenvolvimento elaborado envolve 2 fases: Especificação e Implementação. Na primeira (Especificação) serão definidas, com base nos requisitos, todas as funcionalidades que deverão ser implementadas na segunda fase(Implementação). De seguida apresentamos as tarefas que constituem cada fase, e a forma como estas poderão ser escalonadas e o tempo(em dias) de duração previsto para cada uma:

Especificação

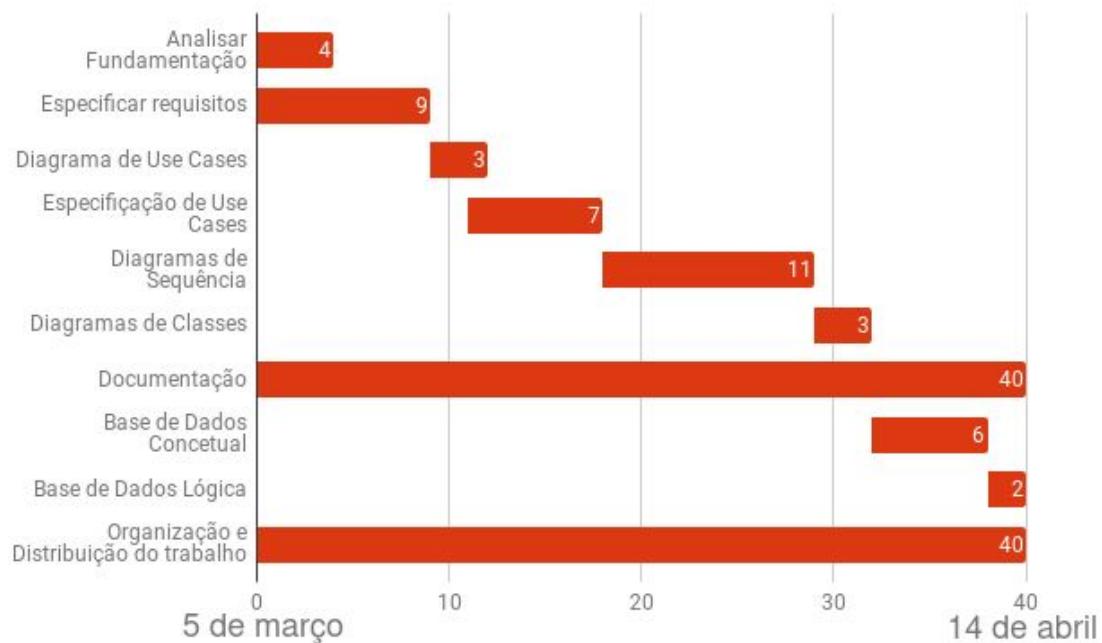


Figura 3 - Diagrama Gantt da especificação

Implementação

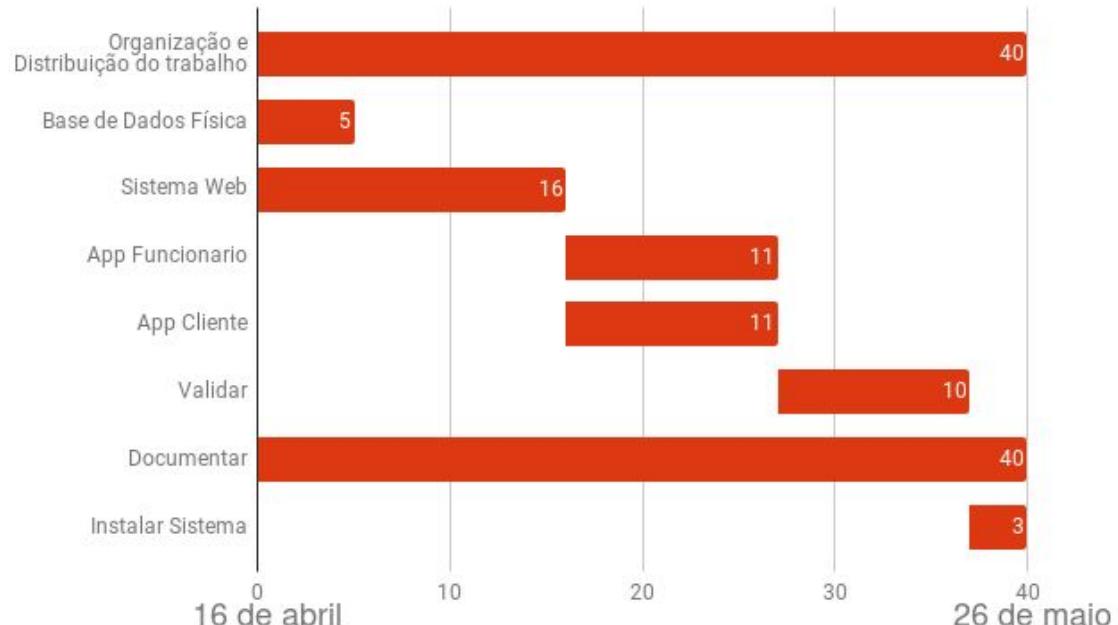


Figura 4 - Diagrama Gantt da implementação

2. Especificação

Como se pode constatar na secção anterior, este projeto apresenta uma interface com o utilizador bem como uma Base de Dados com todos os dados referentes à empresa. Deste modo detetamos a necessidade de seguir o modelo MVC, levando à divisão do projeto em três camadas distintas:

- Camada de Interface(*Presentation*): isola a interface com o utilizador permitindo assim que o resto do projeto não dependa de uma interface concreta, ou seja, poderemos futuramente mudar a mesma.
- Camada de Negócio(*Business*): contém toda a lógica da aplicação.
- Camada de Dados(*Data*): isola o acesso aos dados, de modo a que o resto do projeto não seja dependente da origem ou estrutura sob o qual os dados estão armazenados.

Contudo este modelo obriga à existência de um *Controller* que “ligue” a camada de Interface à Camada de Negócio e que, consoante as ações do utilizador (clicar, escrever, etc), chama as funções da camada negócio correspondentes à ação realizada.

Para a realização deste projeto, decidimos adotar o modelo de desenvolvimento de software em cascata por, na nossa opinião, ser o que mais se adapta ao contexto do problema proposto. O objetivo deste modelo é que as diferentes fases de desenvolvimento de um projeto sigam uma sequência, ou seja, a primeira etapa é direcionada para a segunda, esta mesma para a terceira e assim seguidamente. Na totalidade, este modelo de desenvolvimento comprehende 7 fases distintas, mas no contexto da disciplina em que este projeto se insere apenas iremos realizar as suas 3 primeiras, são elas:

- Análise e Definição dos Requisitos (Requerimento)
- Desenho do Sistema (Especificação)
- Implementação (que será realizada na terceira e última fase deste projeto)

2.1. Requisitos

Após analisar o sistema anteriormente descrito e tendo em conta a arquitetura acima definida o projeto desenvolvido deve ter em conta (respeitar) os seguintes requisitos:

1. Utilizador

a. Funcionais

- i. **Registo de novos utilizadores:** os utilizadores devem ser capazes de se registar na plataforma de modo a terem acesso ao sistema;
- ii. **Alteração de dados:** os utilizadores podem alterar os seus dados (nome,email,password,morada,telefone) pós-registo;
- iii. **Autenticação:** utilizador(**cliente/funcionário**), ao ser redirecionado para a nossa página/aplicação, deve providenciar os dados para autenticação na mesma(email e password);
- iv. **Preenchimento de formulário:** utilizador(**cliente**) preenche formulário de modo a requisitar o envio de uma encomenda, sendo necessário para tal os dados referentes a quem comprou o produto(fornecedor), qual a morada de entrega da encomenda, dados para pagamento e dia e hora a que pretende receber a encomenda;
- v. **Estado da encomenda:** utilizador(**funcionário**) altera estado da encomenda sempre que ou a deixa no centro de distribuição (**com o fornecedor > no centro de distribuição**), ou a recolhe do centro e se encontra em trânsito (**no centro de distribuição > em trânsito**) ou por fim caso a entregue ao destino(cliente) (**em trânsito > entregue**);
- vi. **Avaliação do serviço:** após a entrega da encomenda o utilizador(**cliente**) efetua avaliação do serviço prestado (funcionário que entrega a encomenda) (0 a 10), bem como do estado da encomenda(0 a 5), podendo esta mesma avaliação ser efetuada em ambas as plataformas(web,mobile);
- vii. **Tracking da encomenda:** o utilizador (**cliente**) consulta localização e estado atual da encomenda (**com o fornecedor > no centro de distribuição > em trânsito > entregue**) através da plataforma web/mobile;
- viii. **Consulta da rota:** utilizador(funcionário) consulta a rota mais eficiente até ao fornecedor, até ao centro de distribuição ou até ao destino de entrega (cliente);
- ix. **Consulta do histórico:** o sistema deve permitir aos clientes consultar o histórico de encomendas por eles efetuadas bem como as faturas correspondentes;

b. Não Funcionais

i. De Produto

1. A JJMS deve garantir a entrega da encomenda no horário estabelecido com uma tolerância baseada na distância à central de distribuição (quanto mais longe, maior o tempo de tolerância)

2. Sistema

a. Funcionais

- i. **Registo de novos utilizadores:** registar os dados de novos utilizadores referentes a nome, password, email, morada, telefone na base de dados;
- ii. **Alteração de dados:** alteração dos dados referente ao utilizador já presente na BD;
- iii. **Autenticação:** verifica existência de utilizador na BD, após validação providencia acesso;
- iv. **Preenchimento de formulário:** registar a encomenda na BD para futura delegação da mesma;
- v. **Estado da encomenda:** o sistema deve manter um registo do estado da encomenda até que esta seja entregue, mantendo este registo na base de dados. Este estado é alterado consoante a informação dada pelo utilizador(funcionário), sendo também adicionado o custo associado a movimentos efetuados pela mesma bem como outros custos extra(alfândega, armazenamento);
- vi. **Avaliação do serviço:** atualiza a classificação do funcionário que entrega a encomenda e regista a classificação dada também à encomenda;
- vii. **Tracking da encomenda:** consulta do estado da encomenda na BD;
- viii. **Cálculo da rota:** cálculo da rota desde a posição atual do funcionário até à localização da encomenda (no fornecedor ou na central de distribuição) ou até ao destino da mesma (seja este a central de distribuição ou a morada de entrega)
- ix. **Delegar:** o sistema deve delegar, de modo a reduzir custos e uniformizar a carga de trabalho por funcionário sendo que estes estão distribuídos por zonas, a entrega (ao cliente) e recolha (do fornecedor) de encomendas a funcionário após o registo das mesmas, após a escolha do funcionário deve notificá-lo por email com os dados da encomenda;

- x. **Pagamento:** após a entrega de cada encomenda deve ser efetuado, de maneira automática, o débito do valor correspondente à encomenda, e gerada a correspondente fatura, caso o cliente não possua dinheiro suficiente, é bloqueado, ou seja, não pode efetuar novos requisitos de encomendas;
- xi. Os funcionários podem consultar as rotas bem como alterar o estado da encomenda enquanto que, os clientes podem registar-se, alterar os seus dados pessoais, preencher formulário(requisitar), tracking e avaliar encomenda;
- xii. Informação sobre os fornecedores na BD: necessidade de nome, bem como morada dos fornecedores associados à JJMS
- xiii. Morada do centro de distribuição na BD.

b. Não Funcionais

- i. De Produto
 - 1. Uso de algoritmo que se baseia na quantidade de encomendas que cada funcionário é responsável e na zona a que cada um está atribuído para a delegação de encomendas;
- ii. Organizacionais
 - 1. Uso da framework .NET;
 - 2. Uso do SGBD **SQL Server** tendo em meta a construção de uma BD;
 - 3. Utilização de ferramenta de modelação em UML (**Visual Paradigm**);
 - 4. Normalização da BD;
- iii. Externos
 - 1. Interação com plataformas de fornecedores associados à JJMS por forma a permitir o redirecionamento dos clientes.
 - 2. **Redirecionamento:** após a compra no/ao fornecedor o cliente deve ser redirecionado para página/aplicação da JJMS;

2.2.Use Cases

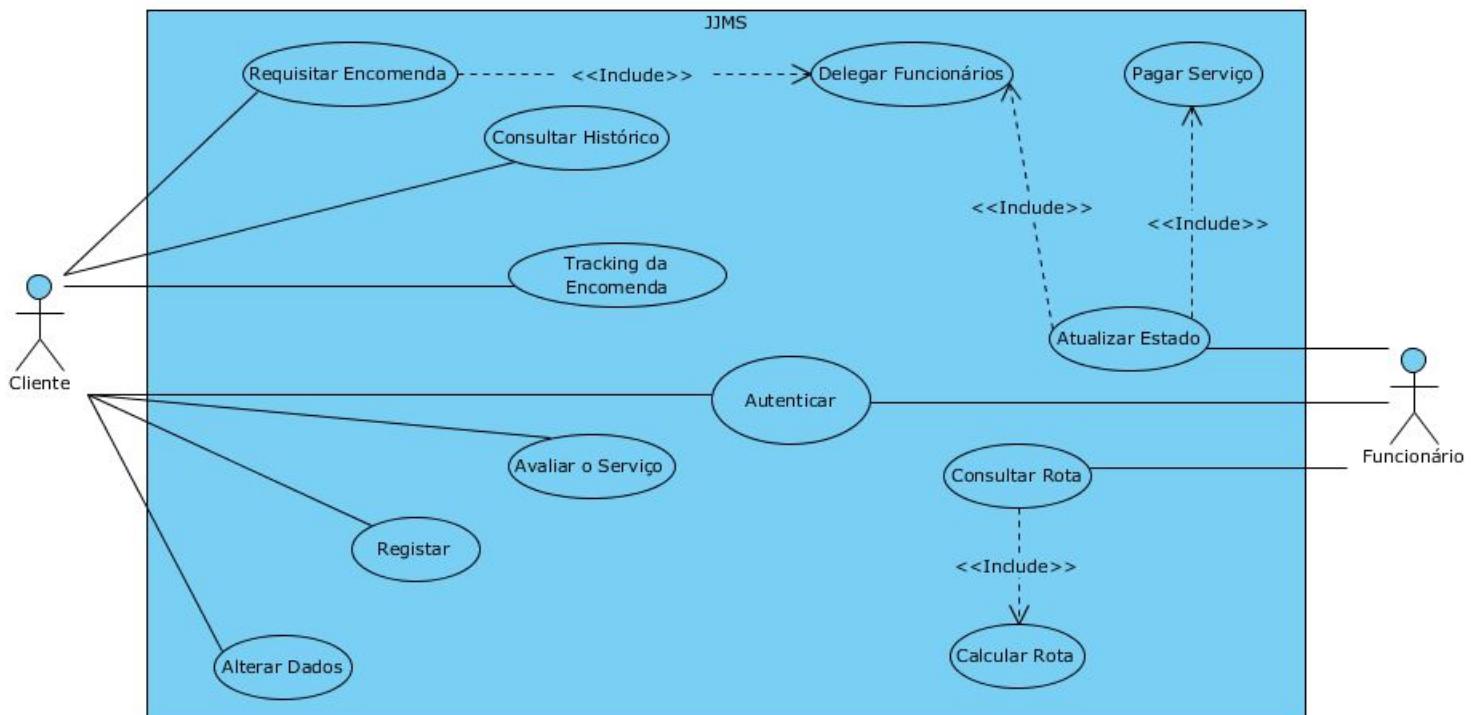


Figura 5 - Diagrama de Use Cases

De modo a explicar o processo usado para a especificação do projeto usaremos, a título de exemplo, o use case “Requisitar Encomenda”, que permite a um cliente requisitar uma encomenda ao nosso sistema, sendo que toda a restante especificação se encontra em anexo neste mesmo documento.

Especificação dos Use Cases:

Para esta fase de desenvolvimento do projeto, consideramos necessário e útil realizar a especificação de todos os use cases definidos, permitindo assim modelar, de uma maneira mais fiel, os procedimentos do sistema e o comportamento do mesmo.

Use Case: Requisitar Encomenda

Pré-condição: Cliente estar autenticado

Pós-condição: Encomenda requisitada com sucesso

	Cliente	Sistema
Comportamento Normal	1. Informa que pretende requisitar encomenda	
		2. Verificar se o cliente não está bloqueado
		3. Pede de que fornecedor é o produto, pede morada de entrega da encomenda e pede dia e hora em que pretende receber a encomenda
	4. Insere nome do fornecedor, morada e dia e hora a receber a encomenda	
		5. Verifica se nome do fornecedor é um dos associados
		6. Pede dados de pagamento
	7. Insere dados de pagamento	
		8. Verifica dados de pagamento
		9. Insere encomenda no sistema
		10. Delegar funcionário para realizar recolha da encomenda ao fornecedor (<<include>> Delegar Funcionários)
Exceção 1 [o cliente está bloqueado] (passo 2)		11. Informa que encomenda foi requisitada com sucesso
		2.1. Informa que o cliente está bloqueado por falta de pagamento de uma encomenda anteriormente realizada
Exceção 2		2.2. Termina sem sucesso
		5.1. Informa que o

[fornecedor inserido não é um dos associados] (passo 5)		fornecedor inserido não é um dos associados à JJMS
		5.2. Termina sem sucesso
Exceção 3 [Dados inválidos] (passo 8)		8.1. Avisa que os dados de pagamento inseridos são inválidos
		8.2. Termina sem sucesso

Tabela 1 - Especificação do use case “Requisitar Encomenda”

Prototipagem dos Use Cases

Sendo a interface, tanto do Browser como da nossa App mobile, uma parte fundamental do projeto final, decidimos “desenhar” possíveis protótipos para a sua aparência, sendo que as interfaces finais deverão estar muito próximas das apresentadas por nós nesta fase. Para além disto, esta prototipagem permite ao programador final ter uma melhor noção do que é pedido e pretendido da parte do cliente, não correndo o risco de implementar a aparência do sistema sem que esta seja aprovada por parte da empresa final.

Requisitar Encomenda

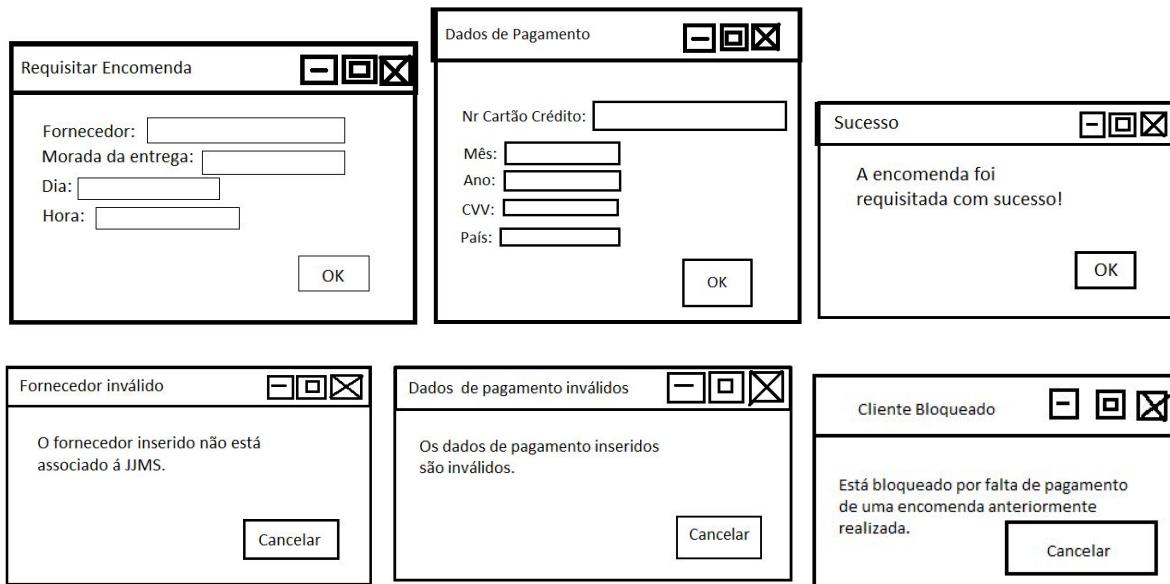


Figura 6 - Prototipagem do “Requisitar Encomenda”

2.3. Diagramas de Sequência

Os diagramas de sequência são diagramas UML que servem para ilustrar a troca de mensagens entre objetos num programa. Este tipo de diagrama coloca em destaque a ordenação temporal destas mensagens, sendo que estas são serviços solicitados de um objecto a outro e as respostas desenvolvidas para as solicitações.

Estas características justificam, portanto, a sua realização para os use case's desenvolvidos.

Desta forma, definimos primeiramente o papel do sistema (use case) e de seguida a forma como software vai desempenhar o seu papel (sequência de operações).

Optamos por dividir o desenho dos diagramas de sequência em três fases distintas:

- Diagramas de sequência dos use case's com conversão literal da especificação dos mesmos
- Diagramas de sequência com imposição de funções/métodos a implementar para tornar possível o uso do use case
- Diagramas de sequência dos métodos enunciados na alínea anterior com conhecimento e desenvolvimento prévio do diagrama de classes utilizado para o projeto

Diagramas de Sequência dos Use Cases (Uses Cases -> Diagramas de Sequência)

Conversão literal da especificação do use case para diagrama de sequência para posterior desenvolvimento do mesmo. Estes diagramas permitem mostrar ao cliente, de forma simples, temporalmente e graficamente cada use case. Visto que o use case “Requisitar Encomenda” inclui o use case “Delegar Funcionários”, apresenta-se aqui também o diagrama de sequência do mesmo.

sd Requisitar Encomenda

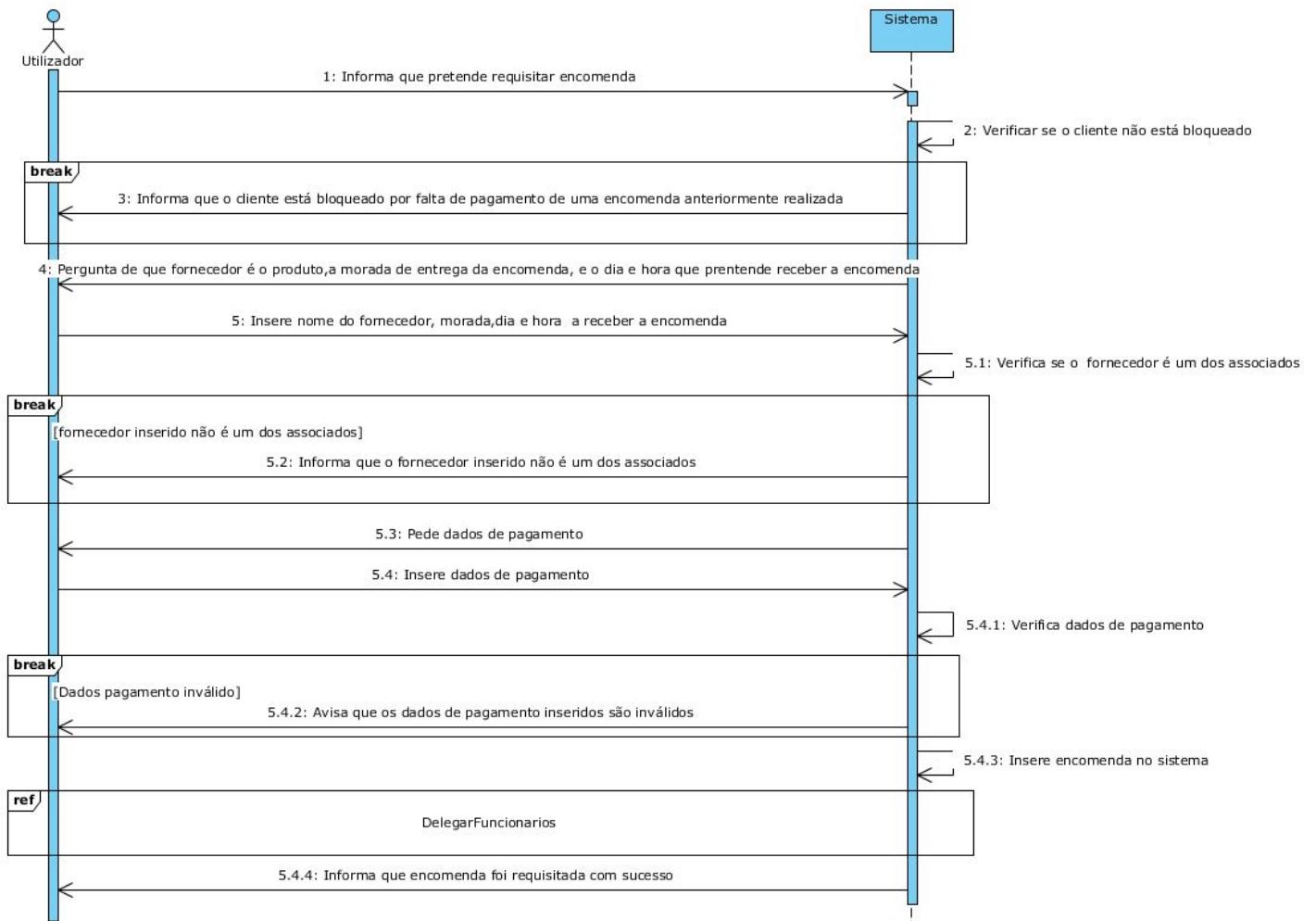


Figura 7 - Diagrama de Sequência de “Requisitar Encomenda”

sd DelegarFuncionarios

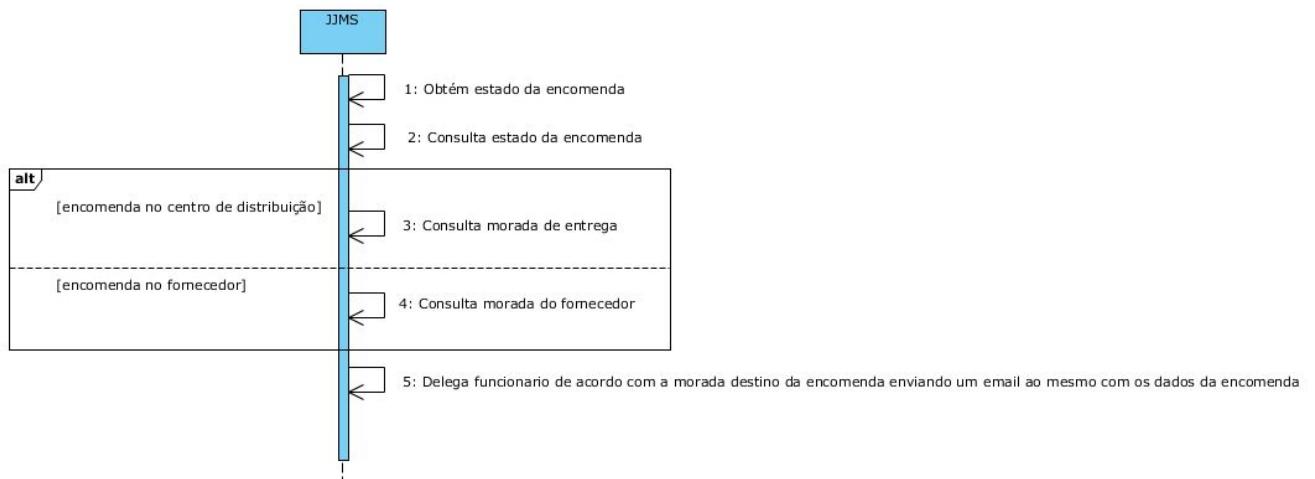


Figura 8 - Diagrama de Sequência de “Delegar Funcionários”

Diagramas de Sequência de Sistema dos Use Cases (Definição de funções)

A partir dos Diagramas de Sequência elaborados anteriormente para cada um dos Use Cases foram gerados os Diagramas de Sequência de Sistema que explicitam a interação entre a camada de interface e a camada de negócio.

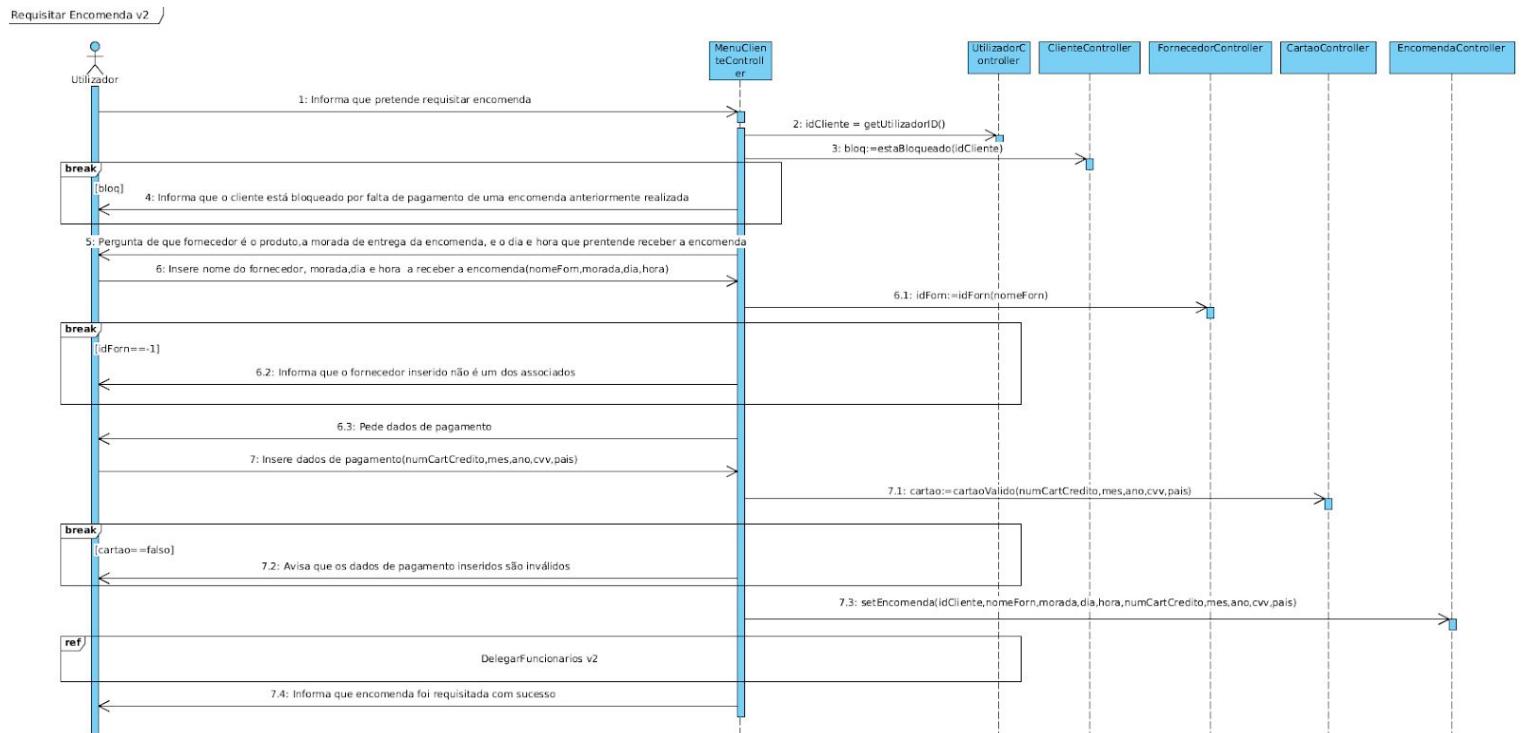


Figura 9 - Diagrama de Sequência de “Requisitar Encomenda” com API

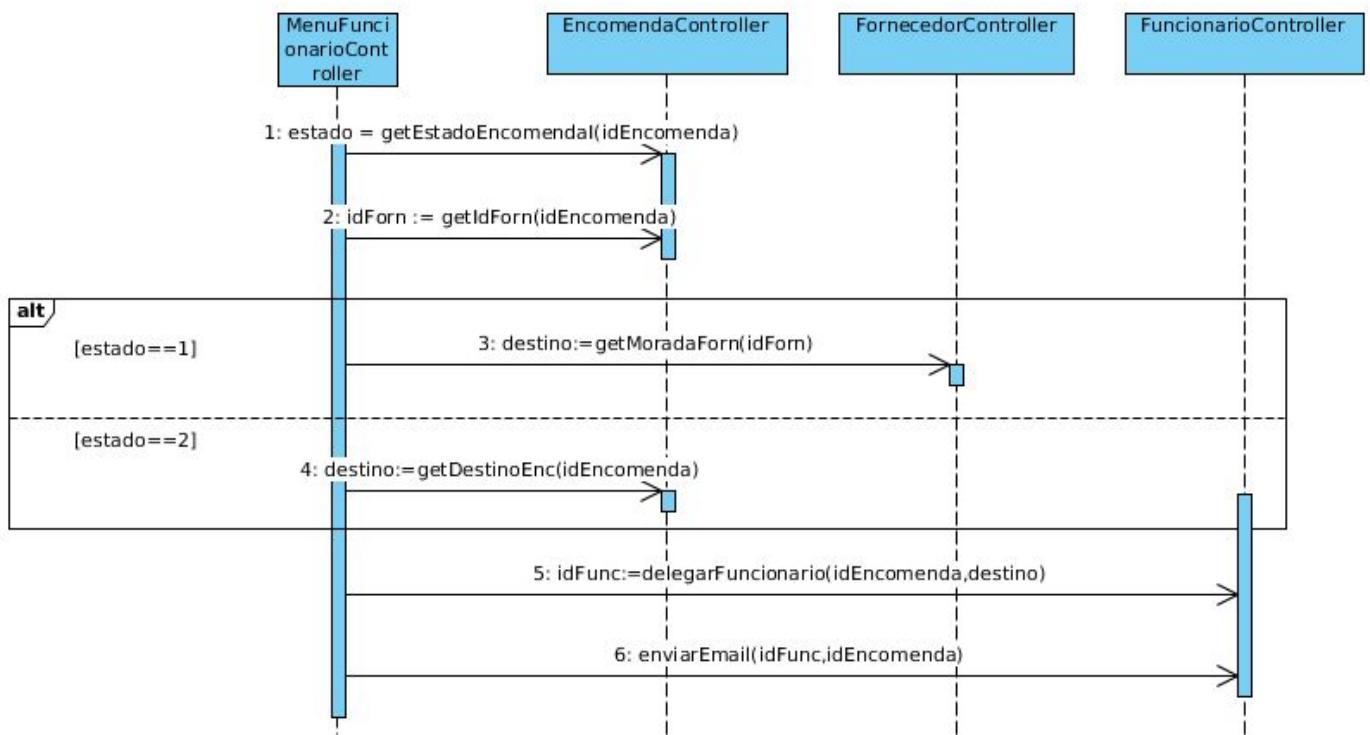


Figura 10 - Diagrama de Sequência de “Delegar Funcionarios” com API

Máquinas de Estado da Prototipagem

O desenvolvimento de máquinas de estado foi motivado pela necessidade de modelar o comportamento da interface a desenvolver agilizando o processo de desenvolvimento da mesma. Foram apenas feitas máquinas de estado para os use cases que possuem interação com o utilizador.

Requisitar Encomenda

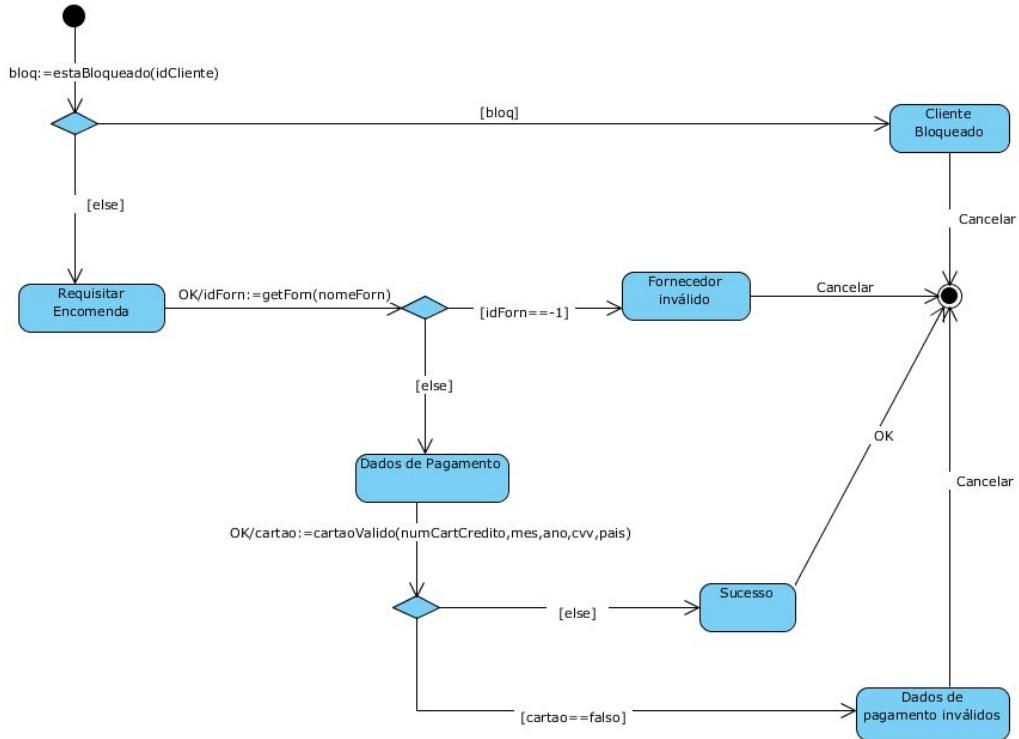


Figura 11 - Máquina de Estados do “Requisitar Encomenda”

Diagramas de Sequência das funções definidas

De modo a facilitar o desenvolvimento do código, e aumentando a granularidade dos diagramas já desenvolvidos, realizámos a especificação das funções anteriormente indicadas através do desenvolvimento dos diagramas de sequência das mesmas, como apresentado de seguida.

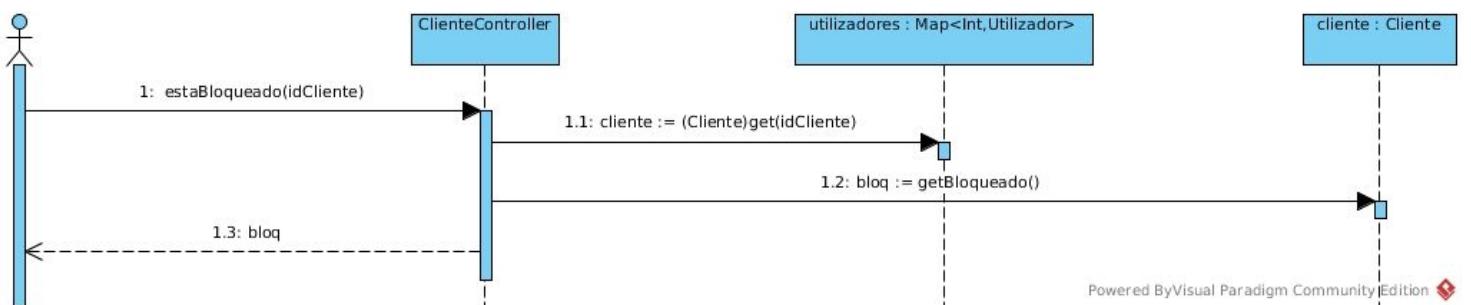


Figura 12 - Diagrama de Sequência de “estaBloqueado”

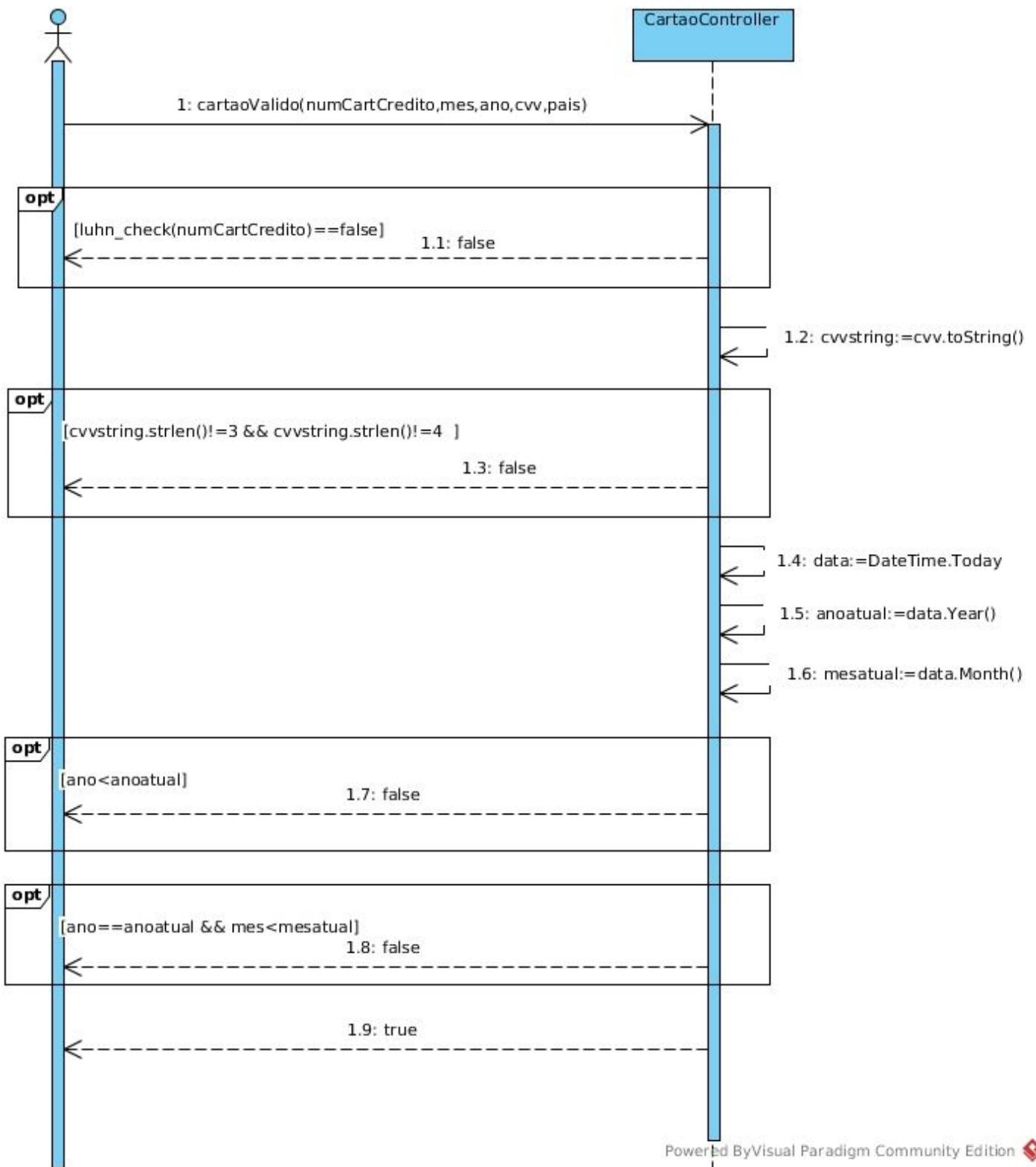


Figura 13 - Diagrama de Sequência de “cartaoValido”

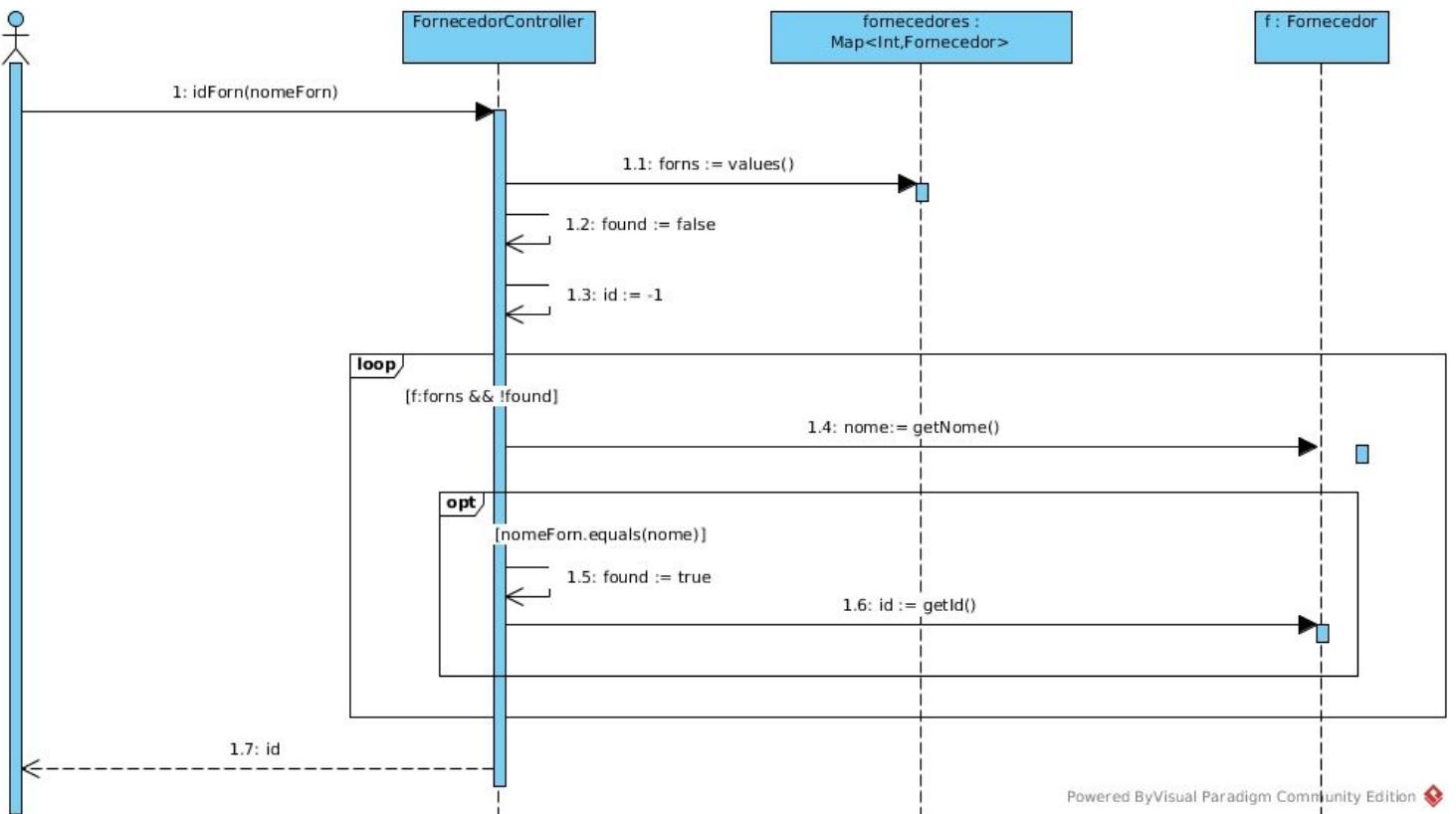


Figura 14 - Diagrama de Sequência de “idForn”

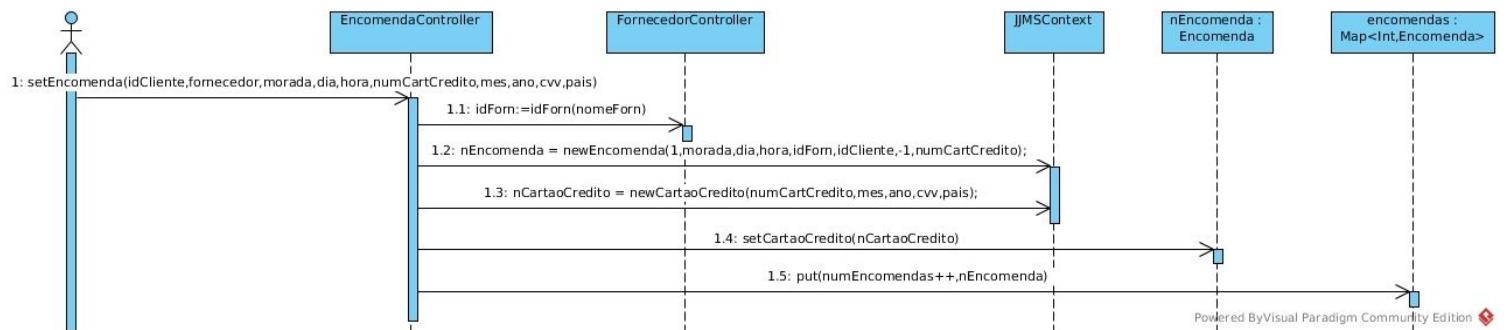


Figura 15 - Diagrama de Sequência de “setEncomenda”

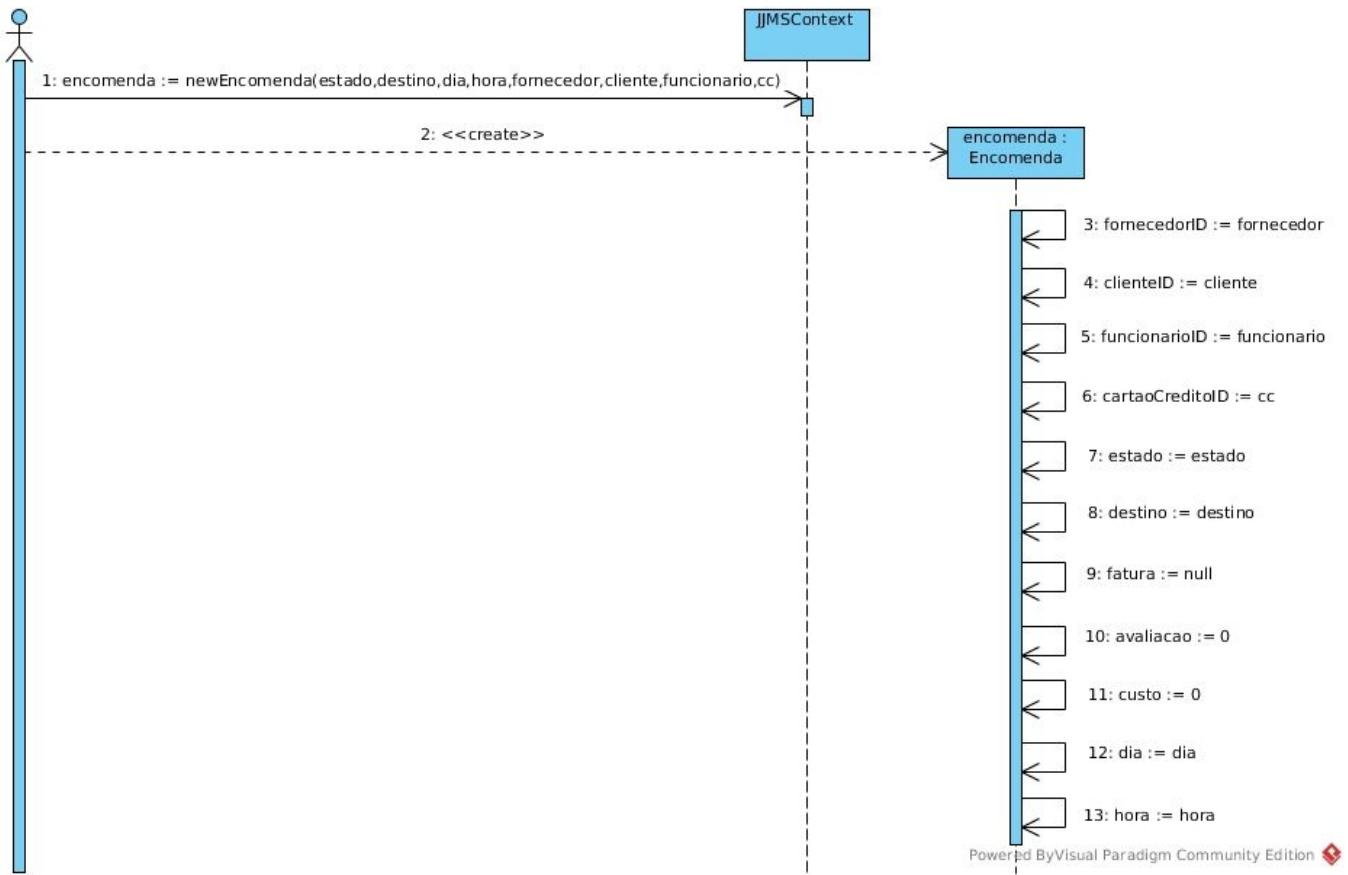


Figura 16 - Diagrama de Sequência de “newEncomenda”

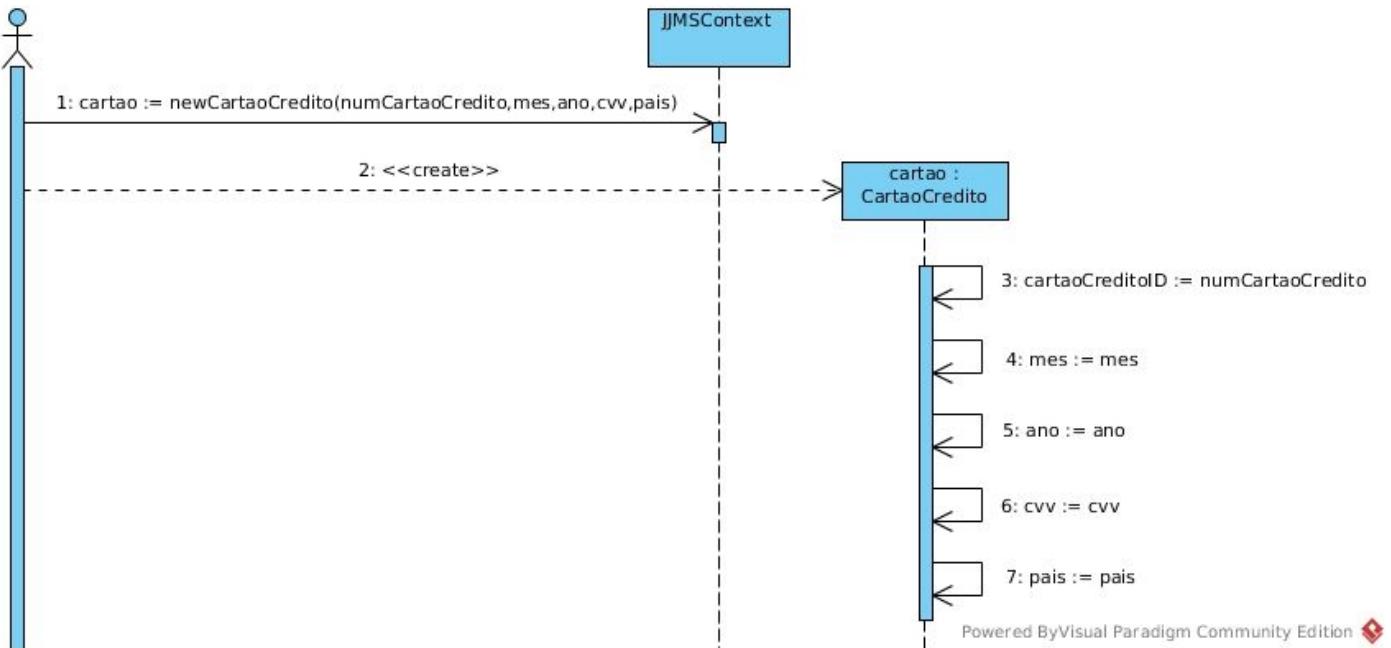


Figura 17 - Diagrama de Sequência de “newCartaoCredito”

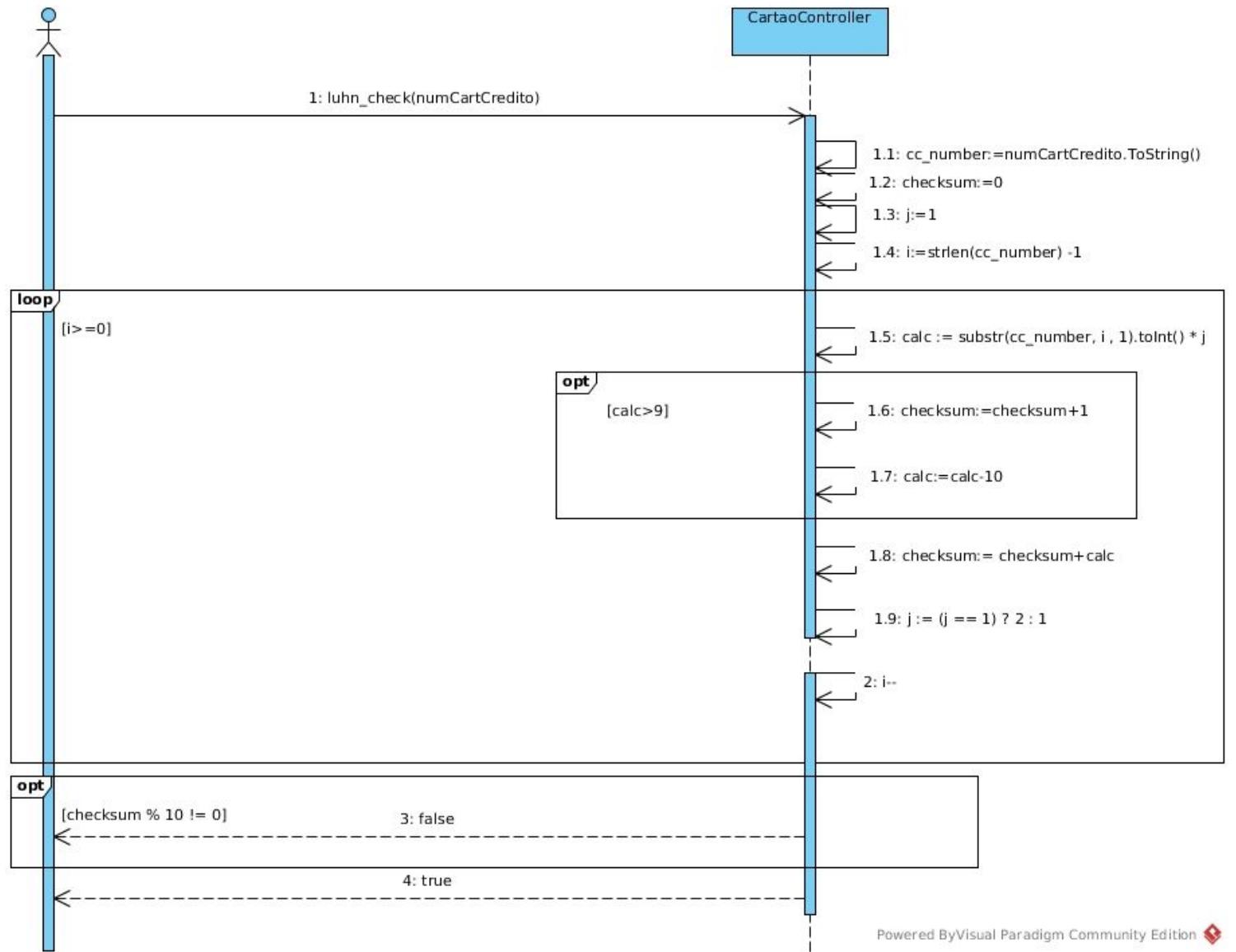


Figura 18 - Diagrama de Sequência de “luhn_check”

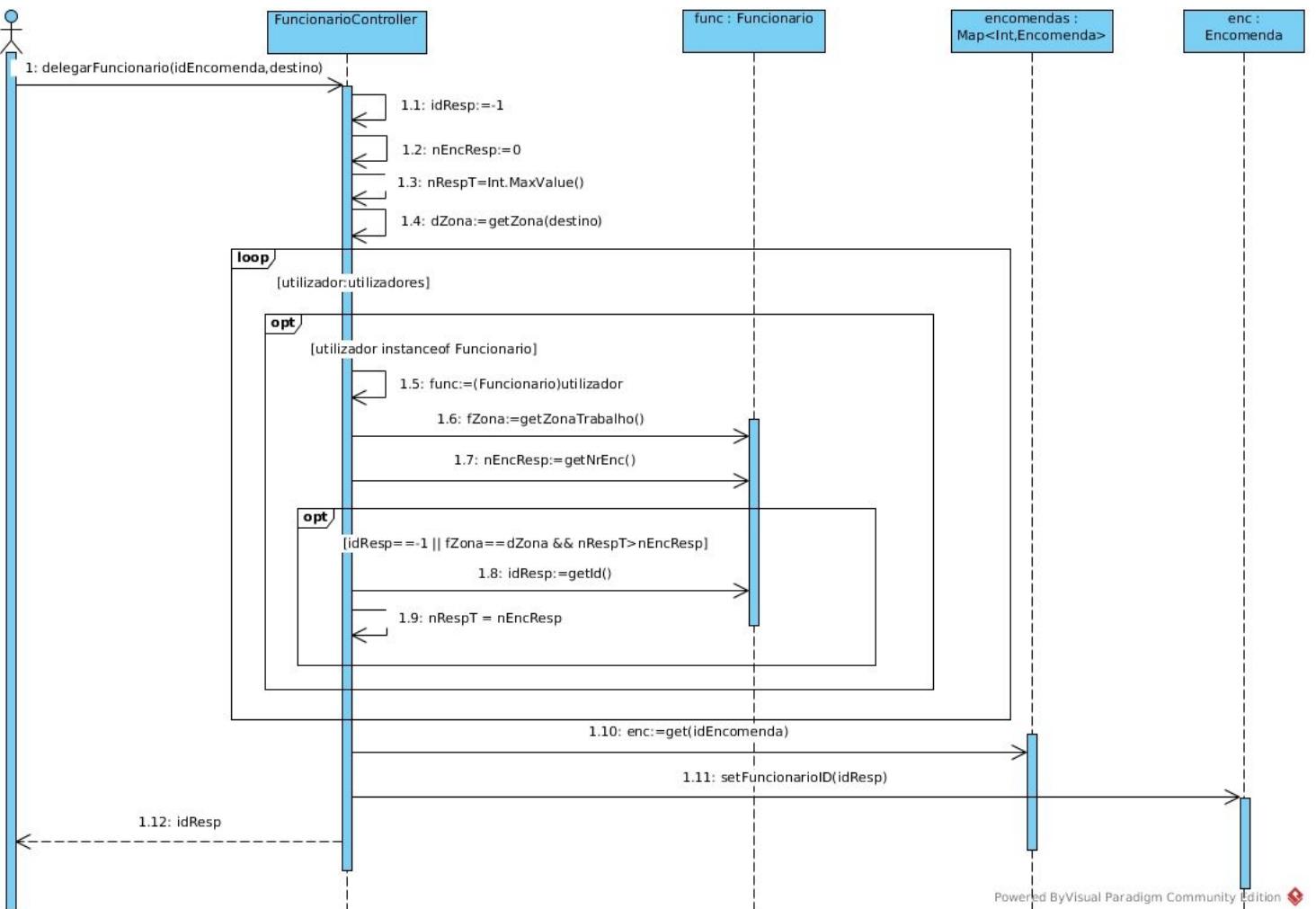


Figura 19 - Diagrama de Sequência de “delegarFuncionario”

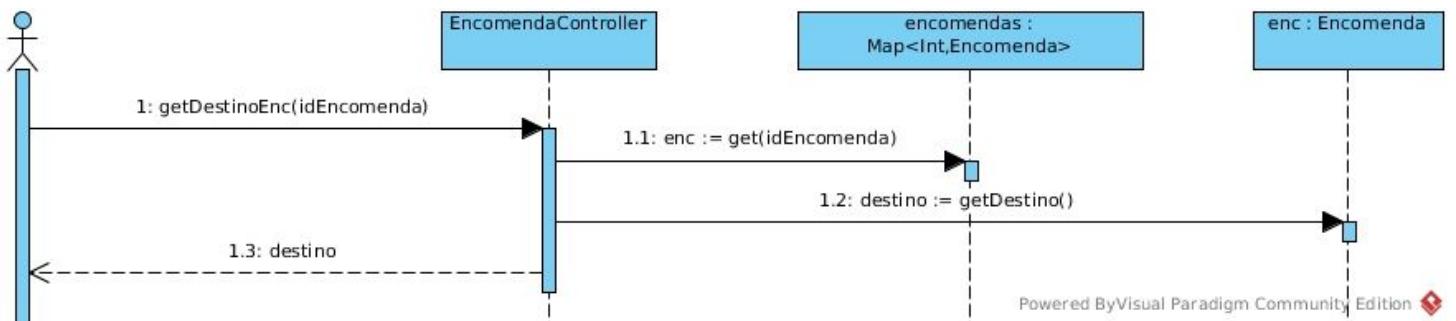


Figura 20 - Diagrama de Sequência de “getDestinoEnc”

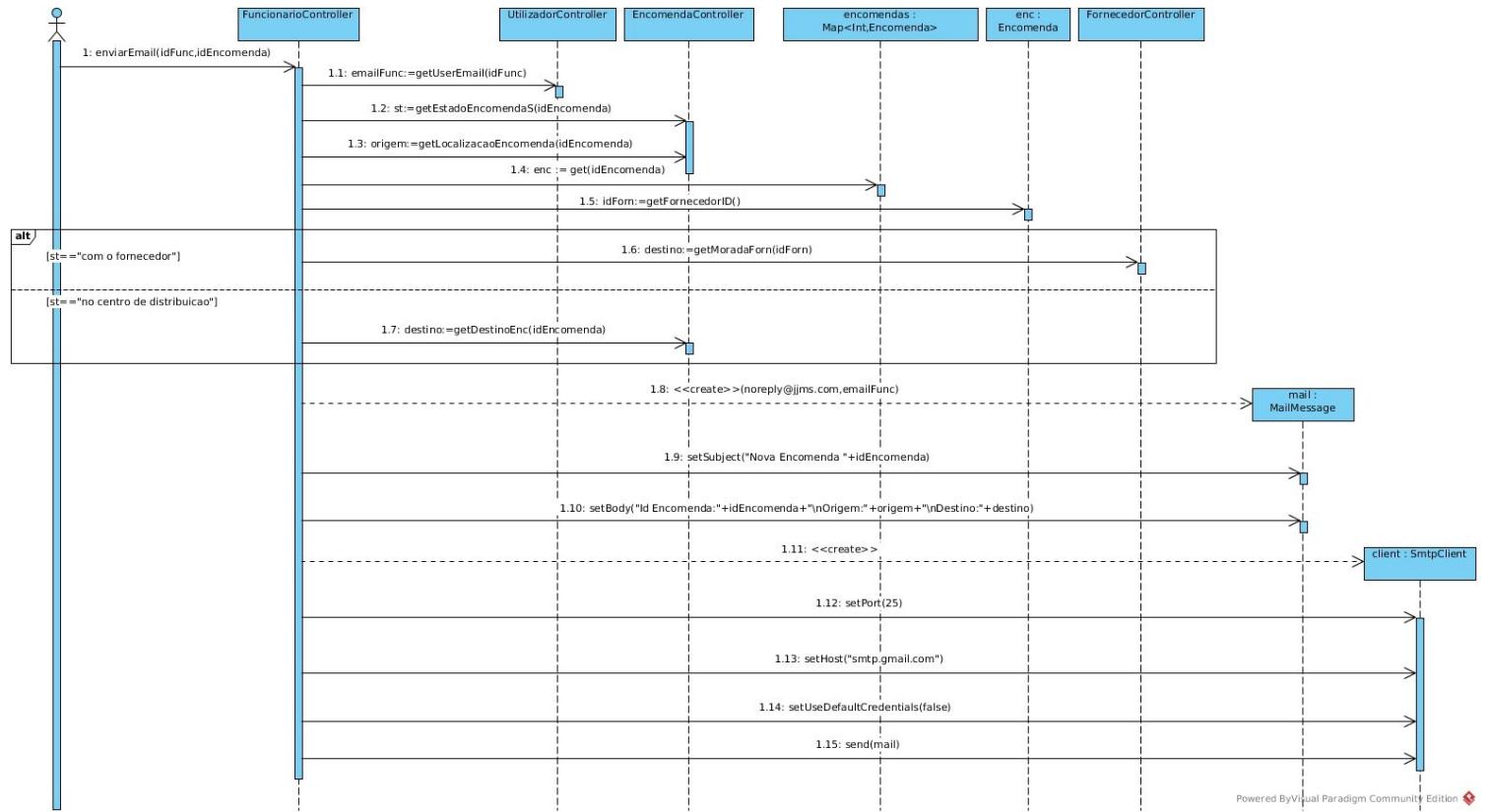


Figura 21 - Diagrama de Sequência de “enviarEmail”

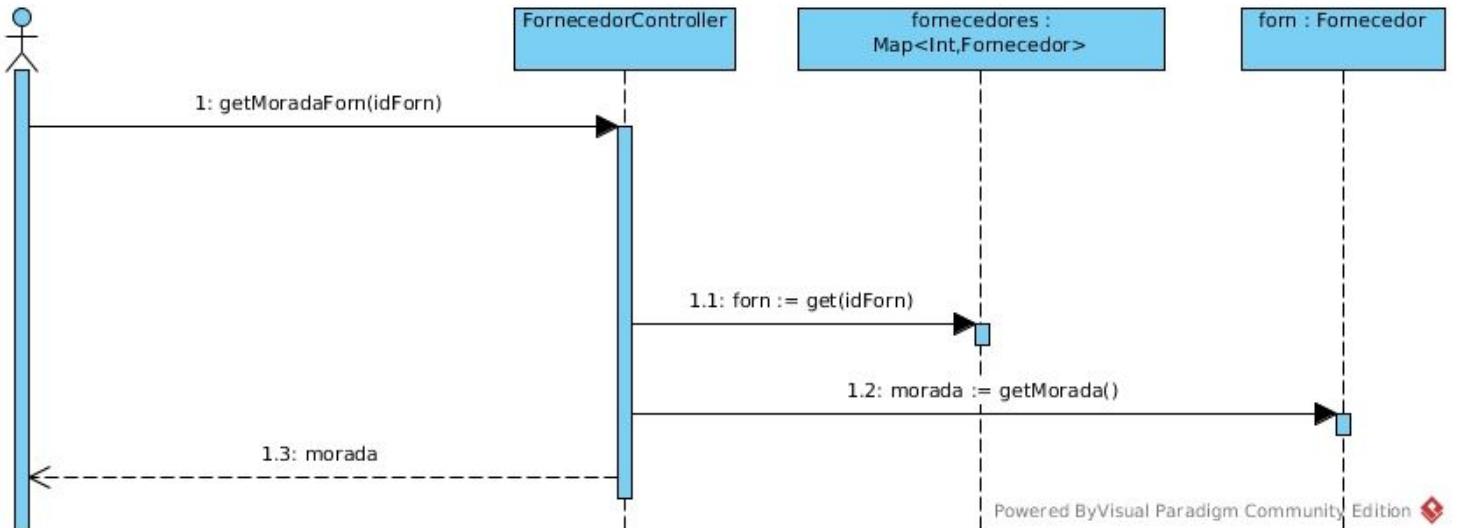


Figura 22 - Diagrama de Sequência de “getMoradaForn”

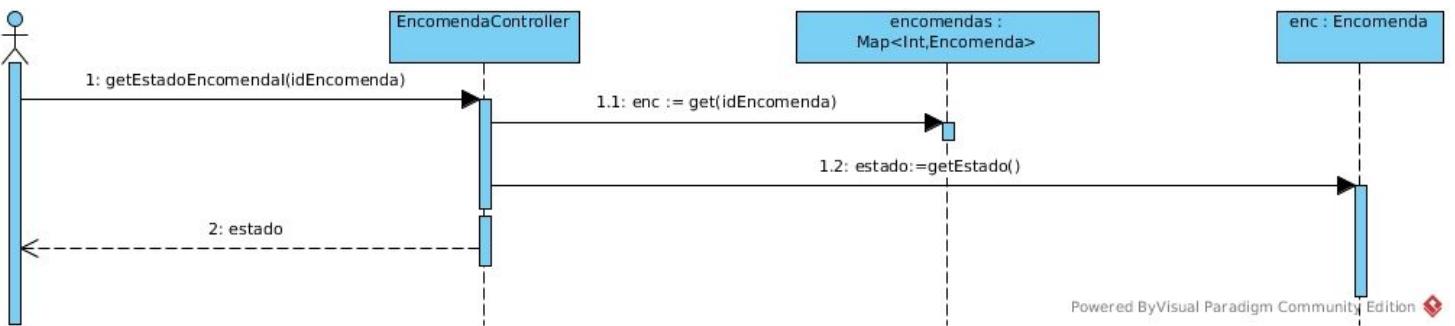


Figura 23 - Diagrama de Sequência de “getEstadoEncomendal”

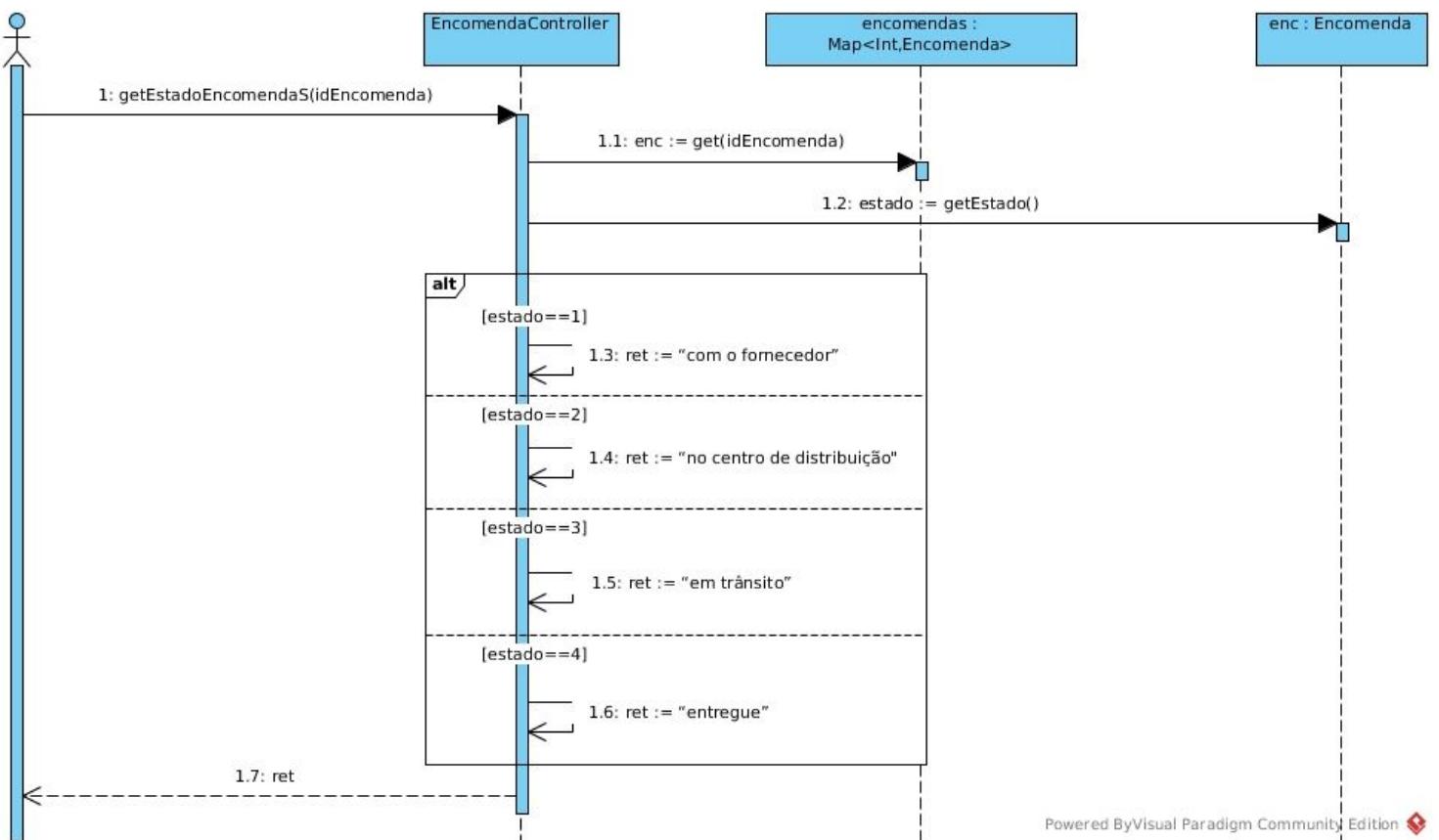


Figura 24 - Diagrama de Sequência de “getEstadoEncomendaS”

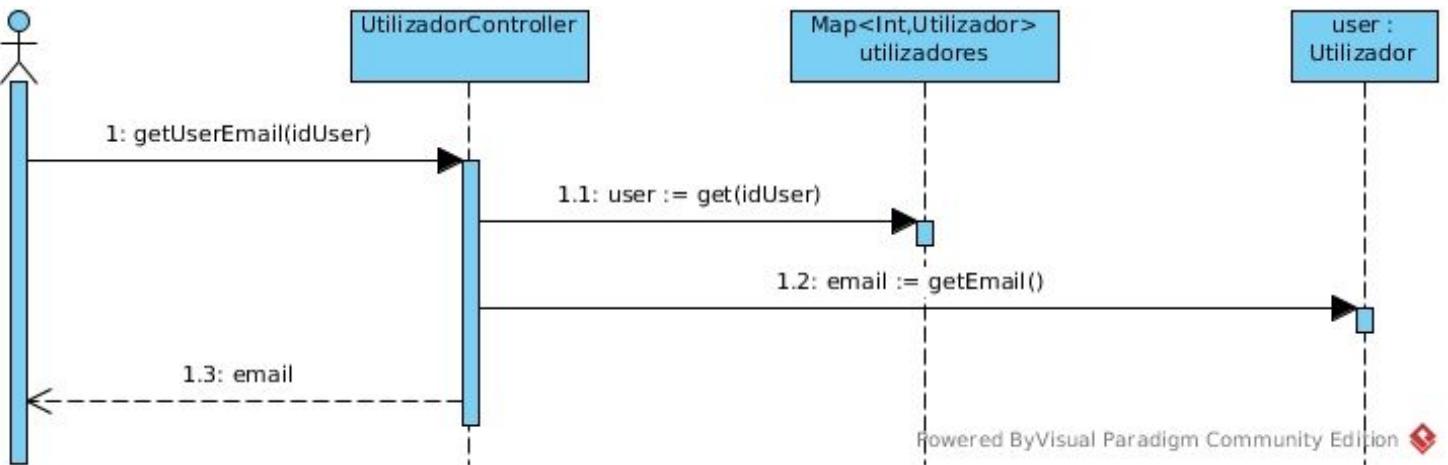


Figura 25 - Diagrama de Sequência de “getUserEmail”

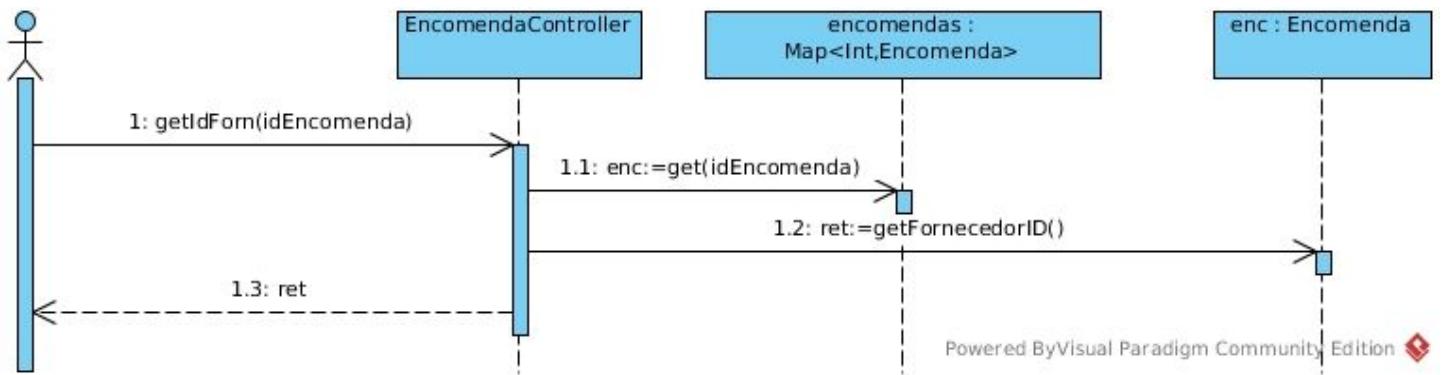


Figura 26 - Diagrama de Sequência de “getIdForn”

2.4. Diagrama de Classes

Após a especificação de grande parte das funções (excluindo a especificação de métodos gets e sets de atributos bem como construtores básicos), definimos o seguinte diagrama de classes:

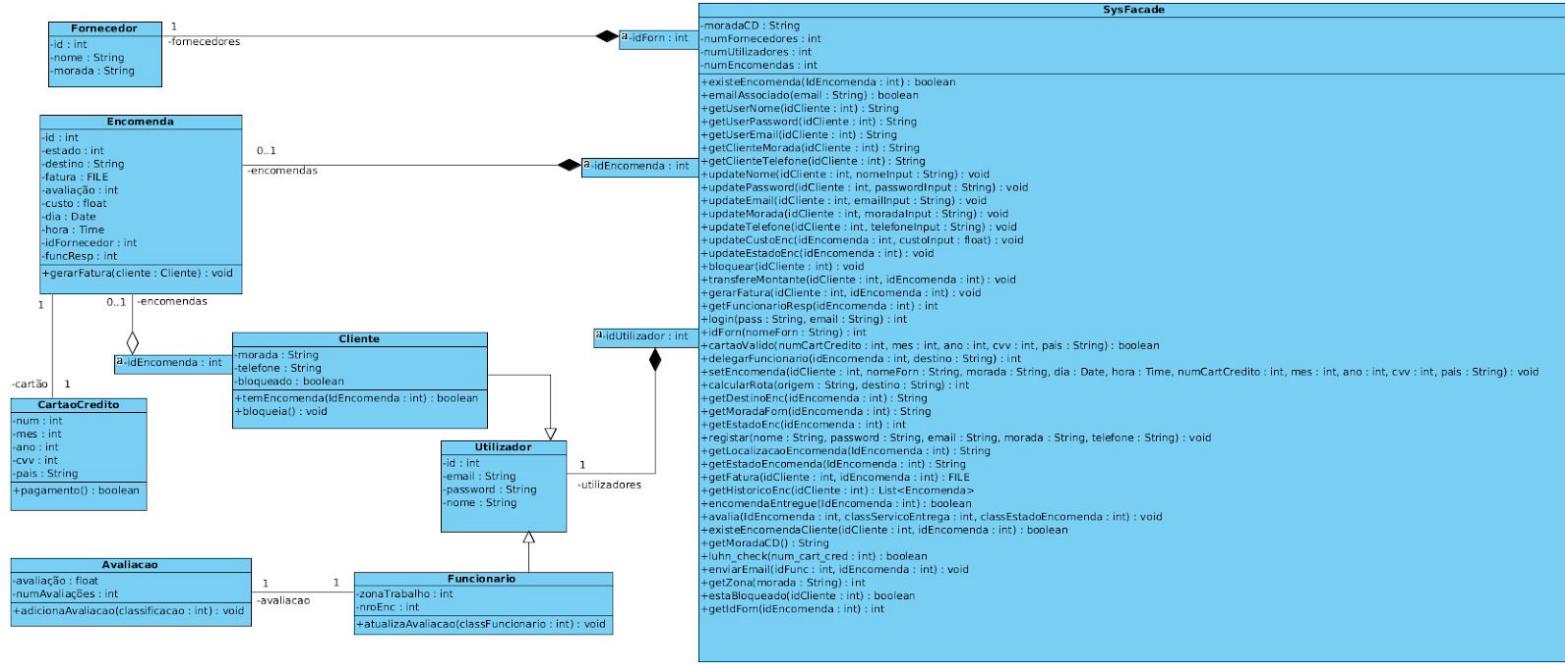


Figura 27 - Diagrama de Classes

Para o desenvolvimento da BD (camada de Dados) decidimos desenvolver o diagrama de classes tendo em conta a persistência dos dados, ou seja, com DAO's visto o mesmo facilitar a criação e justificar escolhas realizadas no desenvolvimento da BD. Apresenta-se de seguida, então o diagrama de classes resultante desta modificação:

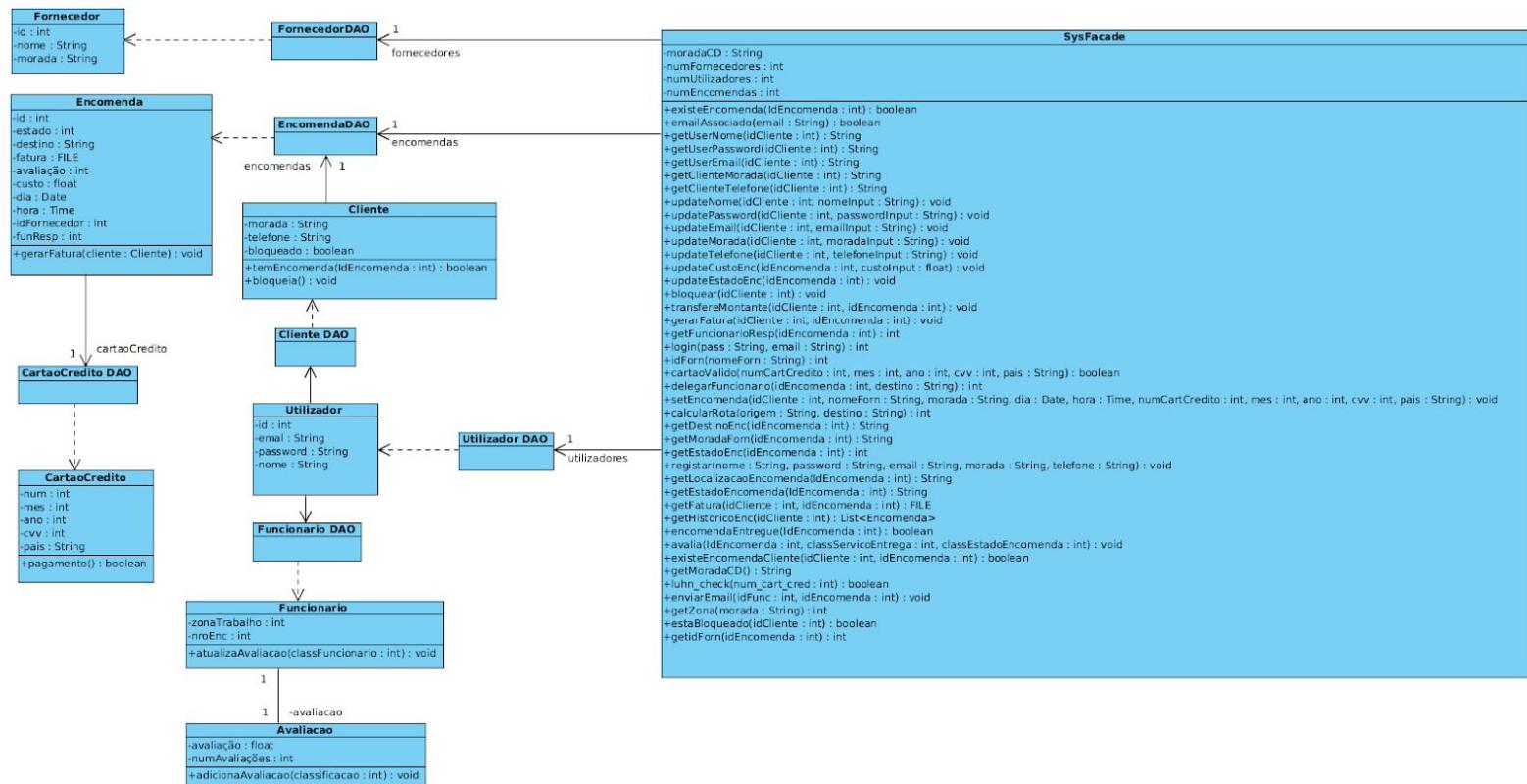


Figura 28 - Diagrama de Classes com persistência

2.5. Modelo (de Base de Dados) Conceptual

Por forma a implementar o sistema desejado, procedemos à esquematização da Base de Dados de suporte ao mesmo, tendo em conta as necessidades funcionais da aplicação e os diagramas de classes base e com DAO's desenvolvidos anteriormente.

Com isto, definimos como entidades da BD clientes e funcionários que são os utilizadores da aplicação (ver multiplicidade das relações estabelecidas, um utilizador ou é um cliente ou um funcionário), encomendas e fornecedores do sistema, e ainda a entidade cartão de crédito para registar a informação dos cartões de crédito utilizados para fazer o pagamento de encomendas da empresa.

A identificação e caracterização das entidades pode ser consultada na seguinte tabela:

Nome da Entidade	Descrição	Alias	Ocorrência
Utilizador	Termo que descreve todos as pessoas que usam o sistema.	User	Cada utilizador pode ser um cliente que requisita o envio de encomendas ou um funcionário que trata da entrega das mesmas.
Funcionário	Termo que descreve as pessoas responsáveis pela recolha de encomendas no fornecedor e da entrega das mesmas no cliente.	Trabalhador	Cada funcionário tem uma zona de trabalho onde efetua o transporte de encomendas.
Cliente	Termo atribuído às pessoas que requisitam o envio de encomendas.	Consumidor	Cada cliente tem impacto no balanço financeiro da empresa, é deles que provêm as receitas.
Encomenda	Termo geral que designa um objecto tratado/transportado pela empresa.	Pacote	Cada encomenda é enviada desde um fornecedor até ao cliente final.
Fornecedor	Termo que designa as empresas que colaboram com a JJMS.	Vendedor	Cada fornecedor da JJMS reencaminha os seus clientes após uma compra na sua empresa para requisitarem o envio da mesma através dos nossos serviços.

			Os funcionários da JJMS tratam de recolher a encomenda no fornecedor.
Cartão Crédito	Termo que designa os dados de pagamento de uma dada encomenda.	Dados de pagamento	Cada cliente que requisita uma encomenda na empresa preenche os dados necessários para que o pagamento do serviço possa ser efetuado com sucesso.

Tabela 2 - Caracterização de Entidades

Por forma a termos uma melhor descrição de cada uma das relações a estabelecer na BD, decidimos criar uma tabela para caracterizar estas relações a definir, sendo que definimos a multiplicidade e descrição de cada uma das relações nos dois sentidos possíveis.

Nome da Entidade	Multiplicidade	Relacionamento	Multiplicidade	Nome da Entidade
Cliente	1	é um	1	Utilizador
Utilizador	1	pode ser	0..1	Cliente
Funcionário	1	é um	1	Utilizador
Utilizador	1	pode ser	0..1	Funcionário
Funcionário	1	transporta	0..N	Encomenda
Encomenda	1	transportada por	1	Funcionário
Cliente	1	requisita	0..N	Encomenda
Encomenda	1	requisitada por	1	Cliente
Encomenda	1	vendida por	1	Fornecedor
Fornecedor	1	vende	0..N	Encomenda
Encomenda	1	tem	1	Cartão Crédito
Cartão Crédito	1	referente a	N	Encomenda

Tabela 3 - Caracterização de Relacionamentos

Por forma a ter um maior conhecimento e demonstrar a utilidade de cada um dos atributos das entidades da BD, a próxima tabela devolve-nos informação sobre cada um destes atributos,

realizando uma descrição do mesmo e mostrando qual o tipo de dados em causa em cada um deles.

Nome da Entidade	Atributos	Descrição	Tipo de Dados e Tamanho	Null's	Multivalor	Alias	Composto	Derivado	Valor Default
Utilizador	Id	Identificador único de cada utilizador	Número inteiro positivo (int)	Não	Não	Número	Não	Não	
	Email	Email de modo a realizar login	Sequência de caracteres (30)	Não	Não	Endereço Eletrónico	Não	Não	""
	Password	Password de modo a realizar login	Sequência de caracteres (48)	Não	Não		Não	Não	""
	Nome	Nome do Utilizador	Sequência de caracteres (25)	Não	Não		Não	Não	""
Funcionário	Id	Identificador único de cada funcionário	Número inteiro positivo (int)	Não	Não	Número	Não	Não	
	zonaTrabalho	Identificador da zona a que o funcionário se encontra atribuído	Número inteiro positivo (int)	Sim	Não		Não	Não	0
	nroEnc	Número de encomendas atualmente delegadas ao funcionário	Número inteiro positivo (int)	Não	Não		Não	Não	0
	avaliação	média de avaliações do funcionário	Número real (float)	Não	Não	Classificação	Sim	Não	0
	numAvaliações	número de avaliações que o funcionário recebeu	Número inteiro positivo (int)	Não	Não		Sim	Não	0
Cliente	Id	Número único que identifica cada cliente	Número inteiro positivo	Não	Não	Id	Não	Não	

			(int)						
	Telefone	Número de telefone do cliente	Sequência de caracteres (15)	Não	Não	Telemóvel	Não	Não	0
	Morada	Morada do cliente	Sequência de caracteres (50)	Não	Não	Residência	Não	Não	""
	Bloqueado	Indica se o cliente possui pagamentos em atraso	Boolean	Não	Não		Não	Não	False
Encomenda	Id	Identificador único para cada encomenda	Número inteiro positivo (int)	Não	Não	Número	Não	Não	
	Estado	Identifica a fase do processo de entrega em que a encomenda se encontra	Número inteiro positivo (int)	Não	Não		Não	Não	1
	Destino	Morada de entrega da encomenda	Sequência de caracteres (50)	Não	Não	Morada de entrega	Não	Não	""
	Fatura	Fatura correspondente ao pagamento da encomenda	Longblob	Sim	Não		Não	Não	
	Custo	Custo total do transporte até ao momento	Número real (float)	Não	Não	Despesa	Não	Não	0
	Avaliação	Avaliação do estado da encomenda aquando a sua entrega	Número inteiro positivo (int)	Sim	Não	Apreciação	Não	Não	0
	Hora	Hora da entrega da encomenda	Time	Não	Não		Não	Não	00:00:00
	Dia	Dia em que a encomenda foi	Date	Não	Não		Não	Não	01-01-01

		entregue							
Fornecedor	id	Identificador único para cada fornecedor	Número inteiro positivo (int)	Não	Não	Número	Não	Não	
	nome	nome do fornecedor	Sequência de caracteres (25)	Não	Não		Não	Não	""
	morada	localização do armazém do fornecedor	Sequência de caracteres (50)	Não	Não	Localização	Não	Não	""
Cartão Crédito	num	Número presente no cartão de crédito	Número inteiro positivo (int)	Não	Não	Id	Não	Não	
	mês	mês presente na validade do cartão	Número inteiro positivo (int)	Não	Não		Não	Não	1
	ano	ano presente na validade do cartão	Número inteiro positivo (int)	Não	Não		Não	Não	1
	cvv	código de segurança do cartão de crédito	Número inteiro positivo (int)	Não	Não	código de segurança	Não	Não	0
	país	nacionalidade do proprietário do cartão	Sequência de caracteres (20)	Não	Não	nação	Não	Não	""

Tabela 4 - Caracterização de Atributos

Para além disto, a informação sobre as chaves de cada uma das 6 entidades da BD a implementar pode ser consultada de seguida:

Entidade	Chave Primária	Chaves Alternativas
Utilizador	id	email
Cliente	id	telefone
Funcionário	id	-
Encomenda	id	-

Fornecedor	id	nome
Cartão Crédito	num	-

Tabela 5 - Caracterização de chaves candidatas e primárias

Tendo em consideração este dicionário, construímos o seguinte modelo ER representando o modelo conceptual da BD:

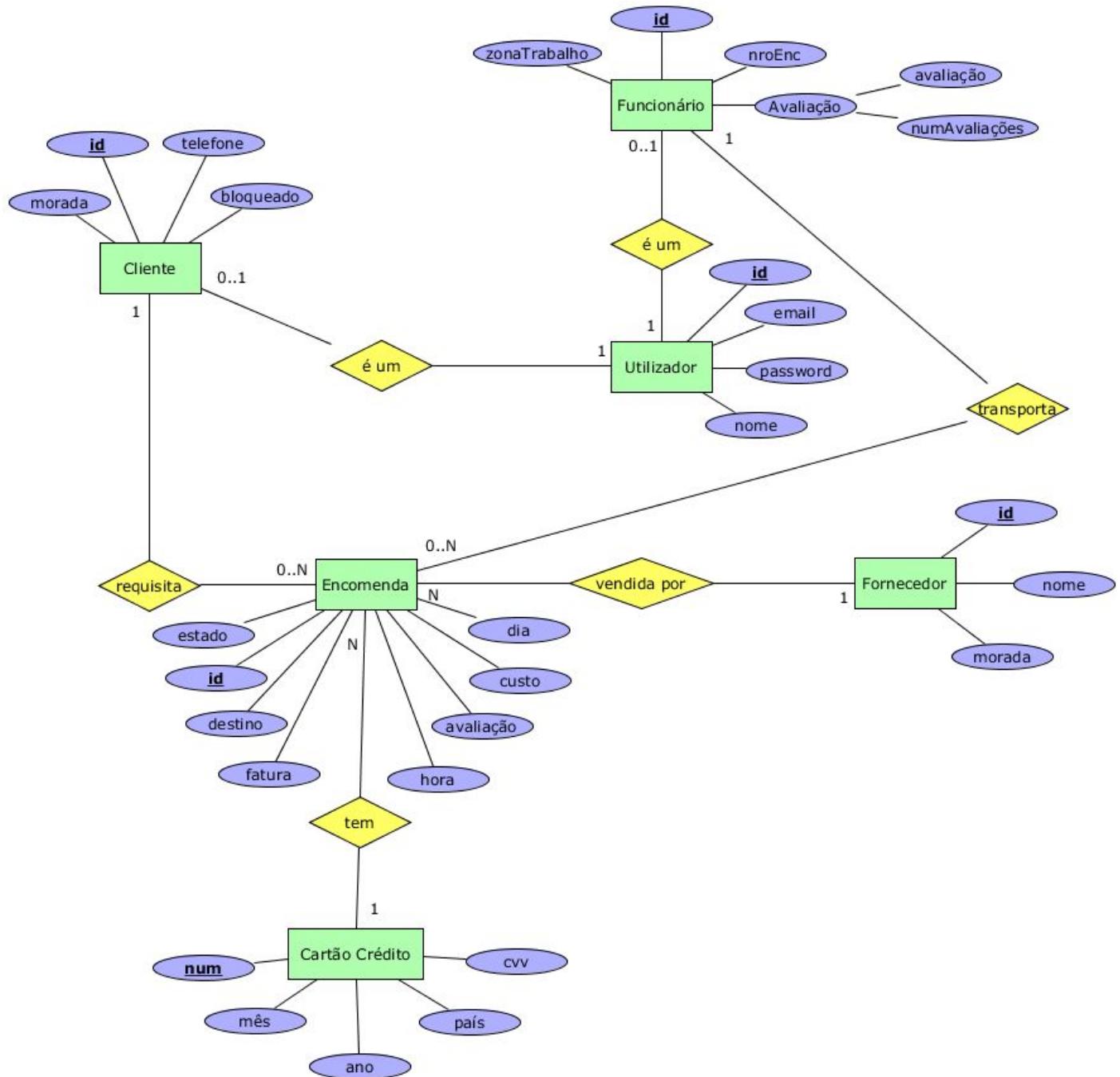


Figura 29 - Modelo ER da BD

2.6. Modelo (de Base de Dados) Lógico

Esta fase de desenvolvimento do projeto termina com a conversão da Base de Dados apresentada em cima (modelo conceptual) para o modelo lógico (que precede a implementação da mesma, a realizar na fase seguinte, modelo físico).

Assim, são convertidas para este modelo todas as entidades para tabelas e declarados o tipo dos atributos que constituem as mesmas, ficando assim cada vez mais próximo do nível de implementação.

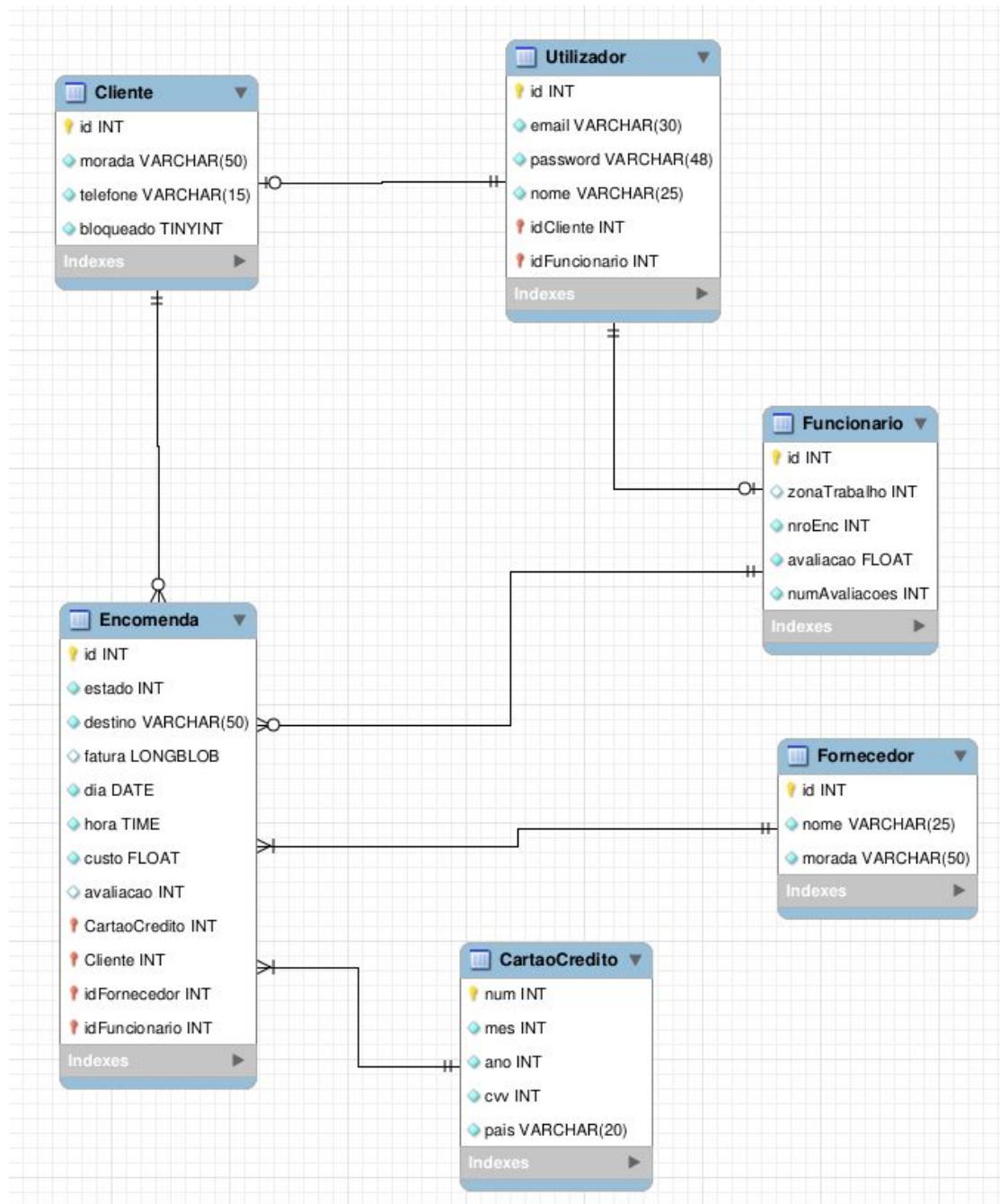


Figura 30 - Modelo Lógico da BD

3. Implementação

A última fase de desenvolvimento consistiu na implementação do sistema definido nas fases anteriores, ainda que com algumas modificações, como poderemos verificar mais à frente. Visto tratar-se de um sistema com duas *front-ends*: *Mobile* e *Web* foi tomada a decisão de usar uma arquitetura **Model-View-Controller**. Foi também tomada a decisão de usar o **Microsoft SQL Server** como sistema de gestão de base dados devido à sua integração de forma transparente com a *framework* **.NET**. O desenvolvimento da aplicação *Web* foi realizada com recurso à *framework* **ASP.NET** que permite desenvolver páginas *Web* dinâmicas para aplicações (e serviços) *Web* possuindo também uma integração transparente com a *framework* utilizada.

3.1. Arquitetura MVC e Organização

Como já foi referido a arquitetura adotada para o sistema desenvolvido foi a de **Model-View-Controller** que permite dividir a aplicação em três camadas distintas, possibilitando o desenvolvimento de cada uma das mesmas de maneira semi-independente. Sendo assim apresentam-se de seguida as classes pertencentes a cada uma das camadas bem como a funcionalidade implementada em cada uma. De realçar que foi desenvolvida uma outra camada, a **Data**, que será (também) apresentada de seguida e que é responsável pela ligação à base de dados.

3.1.1. Models

A camada de dados representa as entidades que serão mapeadas na base de dados pela Entity Framework, como tal foram desenvolvidos os seguintes modelos:

- Utilizador
- Cliente
- Funcionario
- Fornecedor
- CartaodeCredito
- Encomenda

De realçar que a classe Utilizador é uma classe abstracta da qual Cliente e Funcionario são subclasses visto ambos partilharem as propriedades: **Nome,Email,Password**, isto resultará na fusão de ambos numa só relação na base de dados, algo que será explicitado numa secção subsequente. É também importante referir que a classe Avaliacao passa a estar incorporada na classe Funcionario algo que se reflete na estrutura da base de dados em que estes se apresentam como uma única entidade.

3.1.2. Data

A camada **Data** foi implementada de modo a permitir a interação com a base de dados. Esta camada consiste em duas classes:

- DBInitializer
- JJMSContext

A classe **DBInitializer** permite ao sistema verificar se a base de dados se encontra populada e, em caso negativo, realizar a população com alguns dados de teste iniciais.

A classe **JJMSContext** representa a ligação à base de dados, possuindo como atributos objetos do tipo **DbSet< TEntity >** que identificam relações individuais na base de dados:

```
public DbSet<Cliente> Clientes { get; set; }

public DbSet<Funcionario> Funcionarios { get; set; }

public DbSet<Fornecedor> Fornecedores { get; set; }

public DbSet<Encomenda> Encomendas { get; set; }

public DbSet<CartaoCredito> Cartoes { get; set; }

public DbSet<Utilizador> Utilizadores { get; set; }
```

Esta classe permitirá que os controladores interajam com a base de dados, lendo e escrevendo dados na mesma quando necessário, disponibilizando ainda *wrappers* para a criação de objetos de cada uma das relações presentes na base de dados:

```
public Fornecedor newFornecedor(string nome, string morada);

public Cliente newCliente(string nome, byte[] passwordH, string email, string
morada, string telefone);

public Funcionario newFuncionario(string nome, byte[] passwordH, string email,
int zonaTrabalho);

public CartaoCredito newCartaoCredito(long numCartaoCredito, int mes, int ano,
int cvv, string pais);

public Encomenda newEncomenda(int estado, string destino, Date dia, Time
hora, int fornecedor, int cliente, int funcionario, long cc);
```

Por forma a que a base de dados seja criada antes da aplicação começar a correr esta é declarada como um serviço através do mecanismo de **Dependency Injection** disponibilizado pelo **ASP.NET**:

```
services.AddDbContext<JJMSContext>(options =>  
  
    options.UseSqlServer(Configuration.GetConnectionString("DefaultC  
onnection")));
```

especificando ainda a string que contém as credenciais de acesso à base de dados.

3.1.3. Controllers

A implementação da lógica da aplicação está presente na camada de negócio. Os **Controllers** são responsáveis por mediar a interação entre a interface (**View**) e a base de dados (**Models/Data**). Foram portanto desenvolvidos os seguintes controladores:

- MenuPrincipalController
- UtilizadorController
- MenuClienteController
- ClienteController
- MenuFuncionarioController
- FuncionarioController
- FornecedorController
- EncomendaController
- CartaoCreditoController

Os controladores desenvolvidos podem ser divididos em duas categorias distintas de acordo com a funcionalidade implementada:

- controladores responsáveis pela funcionalidade presente nos diferentes menus com os quais o utilizador interage: **MenuPrincipalController**, **MenuClienteController**, **MenuFuncionarioController** ainda que por vezes isto consista num simples wrapper para um método implementado noutro controlador (nomeadamente o Login e o Registo de novos utilizadores no controlador **MenuPrincipalController**).
- controladores responsáveis por gerir, com recurso à classe **JJMSContext**, as entidades/relações presentes na base de dados resultando num mapeamento direto entre controladores(**UtilizadorController**, **ClienteController**, **FuncionarioController**, **FornecedorController**, **EncomendaController**) e modelos/relações.

Os controladores interagem entre si permitindo assim a reutilização de código sendo que isto é conseguido através da declaração dos controladores como serviços através do mecanismo de **Dependency Injection**:

```
services.AddMvc().AddControllersAsServices();
```

3.1.4. Views

O último elemento do sistema corresponde à interface com o utilizador, neste caso a interface é feita via uma aplicação Web. As **Views** são responsáveis por permitir a interação dos utilizadores com o sistema. Sendo assim foram desenvolvidas as seguintes **Views**:

- AlterarDados
- AtualizarEstado
- AvaliarServico
- ConsultarHistorico
- Login
- MenuCliente
- MenuFuncionario
- MenuPrincipal
- Registar
- RequisarEncomenda
- Shared(Partilhadas)
- TrackingEncomenda

Cada uma das vistas apresentadas corresponde a um Use Case definido na fase de especificação (com exceção das vistas **MenuCliente**, **MenuFuncionario**, **MenuPrincipal** e **Shared**) sendo que uma vista pode apresentar diversas *interfaces*. Por exemplo a vista **RequisarEncomenda** possui duas interfaces uma para a escolha da encomenda e agendamento da entrega:

Requisar Encomenda

The screenshot shows a web-based form for selecting and scheduling an order. The form has the following fields:

- Fornecedor:** A text input field.
- Morada da Entrega:** A text input field.
- Dia:** A date input field with the placeholder "mm / dd / yyyy".
- Hora:** A time input field.
- OK:** A button at the bottom left of the form.

Figura 31 -Interface da escolha e agendamento da Encomenda

e outra para a introdução dos dados de pagamento relativos à mesma:

Dados de Pagamento

The screenshot shows a user interface for entering payment details. It consists of five input fields arranged vertically: 'Nr Cartão Crédito:' (Credit Card Number), 'Mês:' (Month) with up/down arrows, 'Ano:' (Year) with up/down arrows, 'CVV:' (CVV) with up/down arrows, and 'País:' (Country). Below these fields is a single 'OK' button.

Figura 32 - Interface da introdução dos dados de pagamento

Para, por exemplo, lidar com as diversas situações anómalas(de erro) como introdução de credenciais inválidos ou introdução de identificadores inexistentes que direcionam o utilizador para um ecrã de erro variando apenas a mensagem(de erro) apresentada foi criada a vista **SimpleMsg.cshtml**:

```
<html>
    <head>
        <meta name="viewport" content="width=device-width" />
        <title>@ViewBag.Title</title>
    </head>
    <body>
        <h1>@ViewBag.Title</h1>
        @ViewBag.Msg<br><br>
        <button type="button"
            onclick="location.href='@Url.Action((string)@ViewBag.Func, (string)
            @ViewBag.File)'>@ViewBag.ButtonName</button>
    </body>
</html>
```

que permite definir uma mensagem personalizada através da variável **ViewBag.Msg**, bem como um título para a janela de erro (**ViewBag.Title**) e também como o nome do botão (**ViewBag.ButtonName**). Visto que diversos casos podem originar esta vista/situação são as variáveis **ViewBag.Func** e **ViewBag.File** que permitem guardar o método (e respetiva classe) que deve ser chamado após a situação de erro.

Os dados introduzidos pelo utilizador são passados aos controladores através do método **HTTP POST**, algo indicado através da diretiva disponibilizada pelo **ASP.NET @Html.Action**:

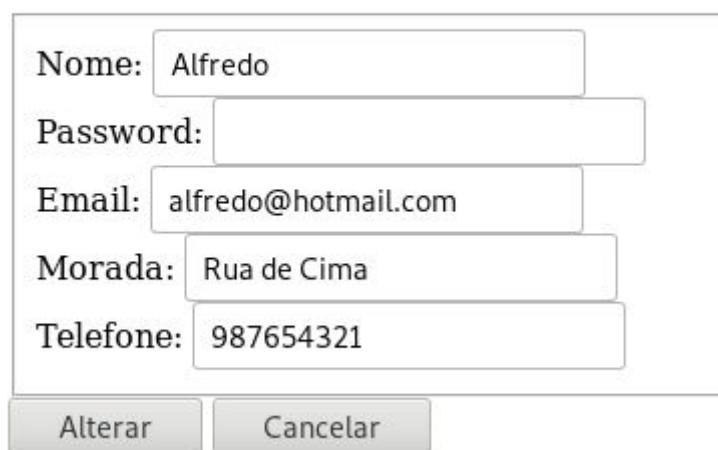
```
@using (Html.BeginForm("Autenticar", "Utilizador", FormMethod.Post))
```

Neste caso os parâmetros requeridos pelo método **Autenticar** da classe **UtilizadorController**, **email** e **password** são passados através do método **HTTP POST**.

Por outro lado, quando um cliente deseja alterar dados referentes à sua conta os seus dados atuais são apresentados na vista com recurso ao objeto **ViewBag**:

```
Nome: <input type="text" name="nomeInput" id="nomeInput"  
value="@ViewBag.nome" required />  
  
<br>  
  
Password: <input type="password" name="passwordInput" id="passwordInput"  
/>  
  
<br>  
  
Email: <input type="text" name="emailInput" id="emailInput"  
value="@ViewBag.email" required />  
  
<br>  
  
Morada: <input type="text" name="moradaInput" id="moradaInput"  
value="@ViewBag.morada" required />  
  
<br>  
  
Telefone: <input type="text" name="telefoneInput" id="telefoneInput"  
value="@ViewBag.telefone" required />
```

Alterar Dados



The screenshot shows a user interface for updating user data. It consists of a form with five input fields and two buttons at the bottom.

- Nome:** Alfredo
- Password:** (Empty)
- Email:** alfredo@hotmail.com
- Morada:** Rua de Cima
- Telefone:** 987654321

At the bottom of the form are two buttons:

- Alterar** (Update)
- Cancelar** (Cancel)

Figura 33 - Interface para alterar os dados

Um aspeto particular do ASP.NET é a funcionalidade que permite apresentar dados armazenados na base de dados de forma automática, sendo esta funcionalidade usada para apresentar as encomendas realizadas por um determinado cliente:

Consultar Histórico

ID da Encomenda	Estado da Encomenda	Morada de entrega	Avaliação	Custo(€)	Dia de entrega	Hora de entrega
1	Com o fornecedor	Quinta da Rua	NA	0	2018-05-28	05:45
1002	Com o fornecedor	Rua das Marinhas	NA	0	2018-06-13	22:00

[Voltar](#)

Figura 34 - Interface para consultar histórico

3.2. EntityFramework usada com Code First Approach

A implementação da camada de dados foi realizada com recurso à **EntityFramework** com uma abordagem designada **Code First**. Esta abordagem é caracterizada pelo facto da base de dados ser modelada a partir das classes definidas, sendo de seguida criada com recurso a uma classe que seja do tipo **Context** que representa uma conexão ao sistema de gestão de base de dados. Este, quando é criado, verifica se existe uma base de dados com a estrutura definida e, caso não exista, procede à sua criação e população. Sendo assim resulta desta abordagem o seguinte modelo lógico:

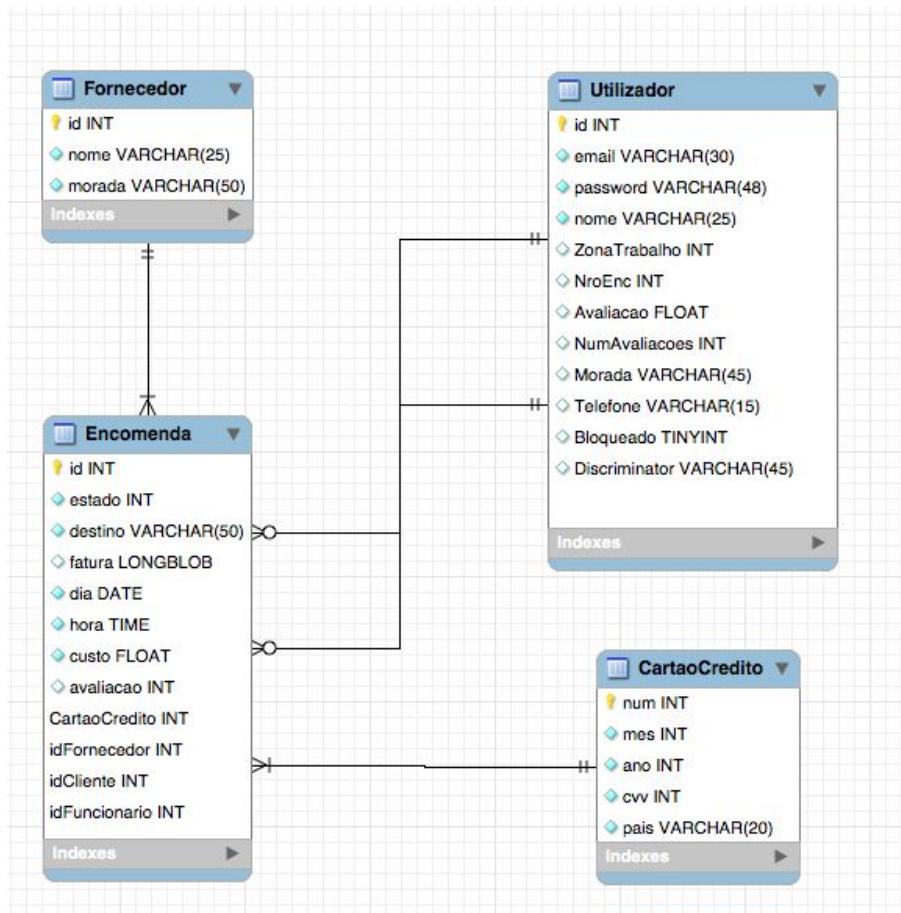


Figura 35 - Modelo Lógico com EntityFramework e Code First

Visto ser esta a abordagem levada a cabo é de realçar as mudanças sofridas pelo modelo lógico desenvolvido na fase de especificação nomeadamente a fusão das tabelas de **Cliente** e **Funcionario** na tabela **Utilizador** que foi motivada pela estratégia de mapeamento de classes em tabelas numa base de dados designada **Table-Per-Hierarchy**. De realçar também que, a multiplicidade das relações bem como a sua obrigatoriedade é conseguida através da definição de atributos tendo em conta a sintaxe definida pela EntityFramework para esse efeito. A título de exemplo apresenta-se de seguida a classe que modela a entidade Encomenda:

```

public int EncomendaID{get;set;}
public int estado{get;set;}
public string destino{get;set;}
public FILE fatura{get;set;}
public int avaliação{get;set;}
public float custo{get;set;}
public Date dia{get;set;}
public Time hora{get;set;}
[ForeignKey("Fornecedor")]
public int FornecedorID;

```

```

public Fornecedor Fornecedor{get;set;}
[ForeignKey("Cliente")]
public int ClienteID;
public Cliente Cliente{get;set;}
[ForeignKey("Funcionario")]
public int FuncionarioID;
public Funcionario Funcionario{get;set;}
[ForeignKey("CartaoCredito")]
public long CartaoCreditoID;
public CartaoCredito CartaoCredito{get;set;}

```

A sintaxe definida pela EntityFramework designa as seguintes restrições/regras para os atributos de entidades que serão posteriormente mapeadas em relações numa base de dados:

- **Chaves primárias:** devem ser atributos com o seguinte nome: **ID** ou **NomeDeClasseID**
- **Chaves Estrangeiras:** devem seguir a mesma estrutura que as chaves primárias e, adicionalmente, devem conter uma “Propriedade de Navegação” i.e um atributo do tipo da classe a que se refere a chave estrangeira

Seguindo estas diretrizes é possível obter uma base de dados com a estrutura planeada de forma automática. Como tal, e voltando ao exemplo da entidade Encomenda, realça-se o atributo que será usado como chave primária: **EncomendaID**, bem como as três chaves estrangeiras para as entidades Cliente(**ClienteID**), Funcionario(**FuncionarioID**), CartaoCredito(**CartaoCreditoID**) e respetivas “Propriedades de Navegabilidade”: **Cliente**, **Funcionario**, **CartaoCredito**.

Visto que a multiplicidade desta relação (**Encomenda-Cliente** e **Encomenda-Funcionario**) é 1-N é necessário a criação de um atributo do tipo **ICollection<Encomenda>** em cada uma destas classes (**Cliente** e **Funcionario**):

```
public ICollection<Encomenda> Encomendas { get; set; }
```

3.2.1. Table per Hierarchy para sistema de subclasses

Uma das estratégias do **EntityFramework** para mapear as classes para a base de dados é a **Table per Hierarchy**. Esta abordagem agrupa as subclasses de uma dada superclasse numa só tabela com o nome da superclasse, recorrendo à coluna **Discriminator** para distinguir as subclasses.

Sendo assim a tabela final irá possuir todas as colunas tanto das subclasses bem como da superclasse sendo que parte dos valores das colunas serão nulos dependendo da subclass

que cada linha pretende representar, ou seja os valores que não lhe pertencem nem pertençam à superclasse serão nulos. No nosso trabalho um exemplo disto pode ser:

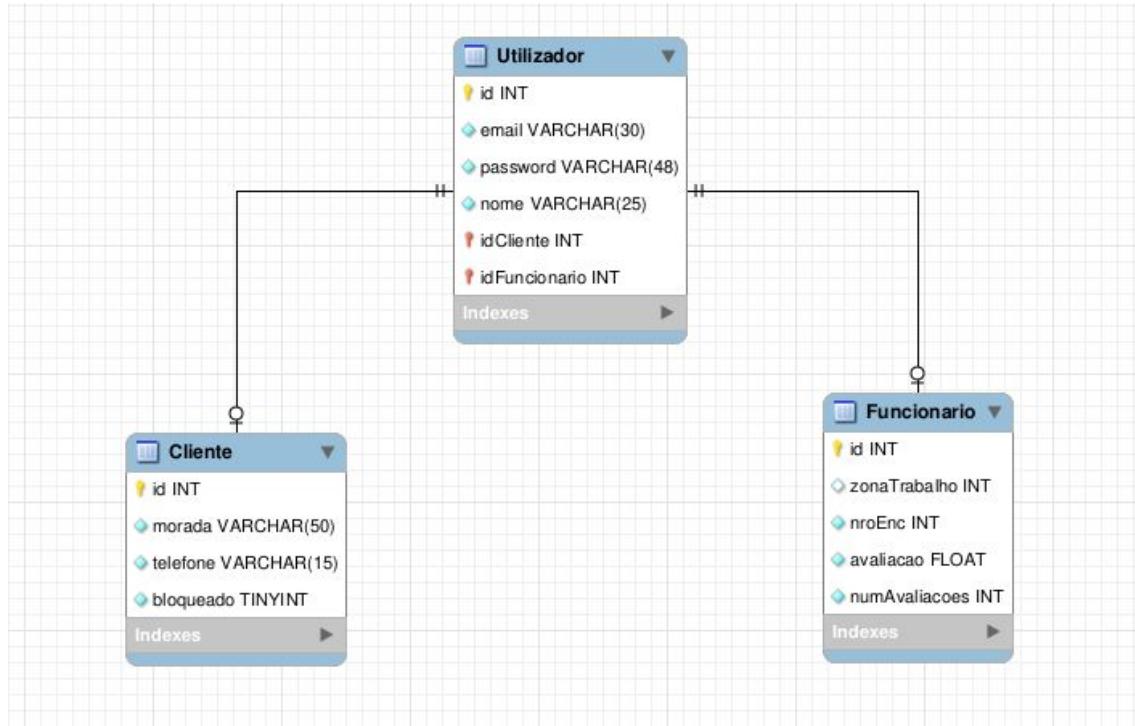


Figura 36 - Tabelas antes

que resulta em:

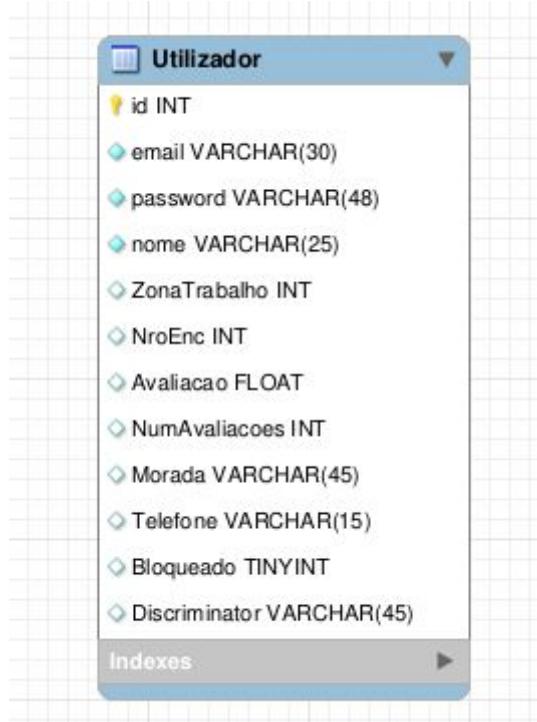


Figura 37 - Tabelas após

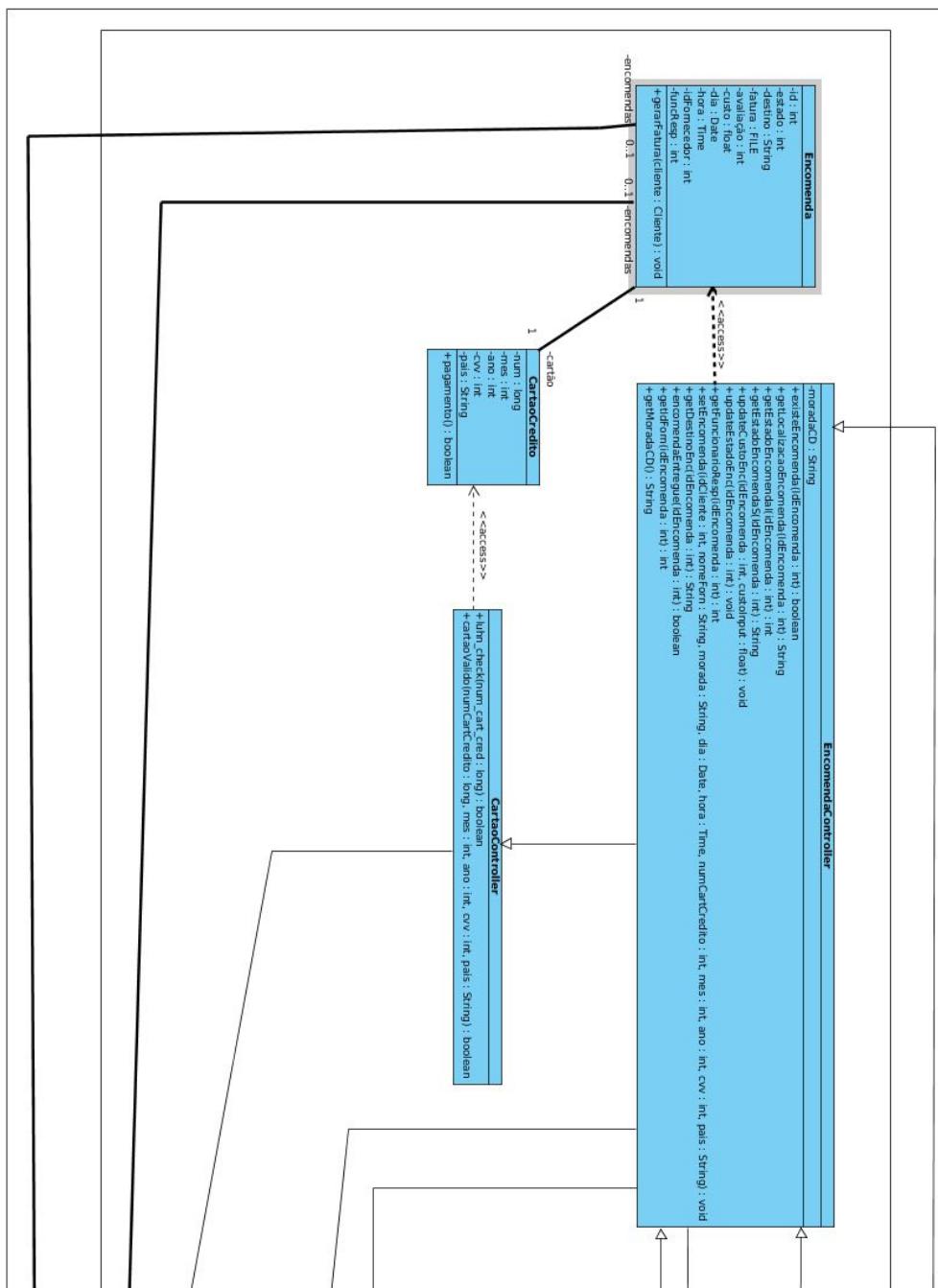
Esta abordagem é mais simples e apresenta melhor performance em relação às alternativas(**Table-per-Type** e **Table-per-Concrete-class**). Contudo esta abordagem leva à

violação da terceira forma normal (existem dependências funcionais entre colunas que não são chave), resultando numa BD não normalizada.

3.3. Adaptações à especificação

A fase anterior resultou no desenvolvimento de uma especificação (formal) do sistema a desenvolver com vista a facilitar a implementação do mesmo. A fase de implementação envolveu, no entanto, algumas mudanças na estrutura do sistema desenhado e, como tal, estas mudanças refletiram-se na especificação anteriormente desenvolvida nomeadamente no diagrama de classes (tanto com como sem persistência). Esta mudança foi motivada pela arquitetura **MVC** que não tinha sido em conta anteriormente e que mudou o modelo de classes apresentado anteriormente de forma significativa.

Diagrama de Classes sem persistência



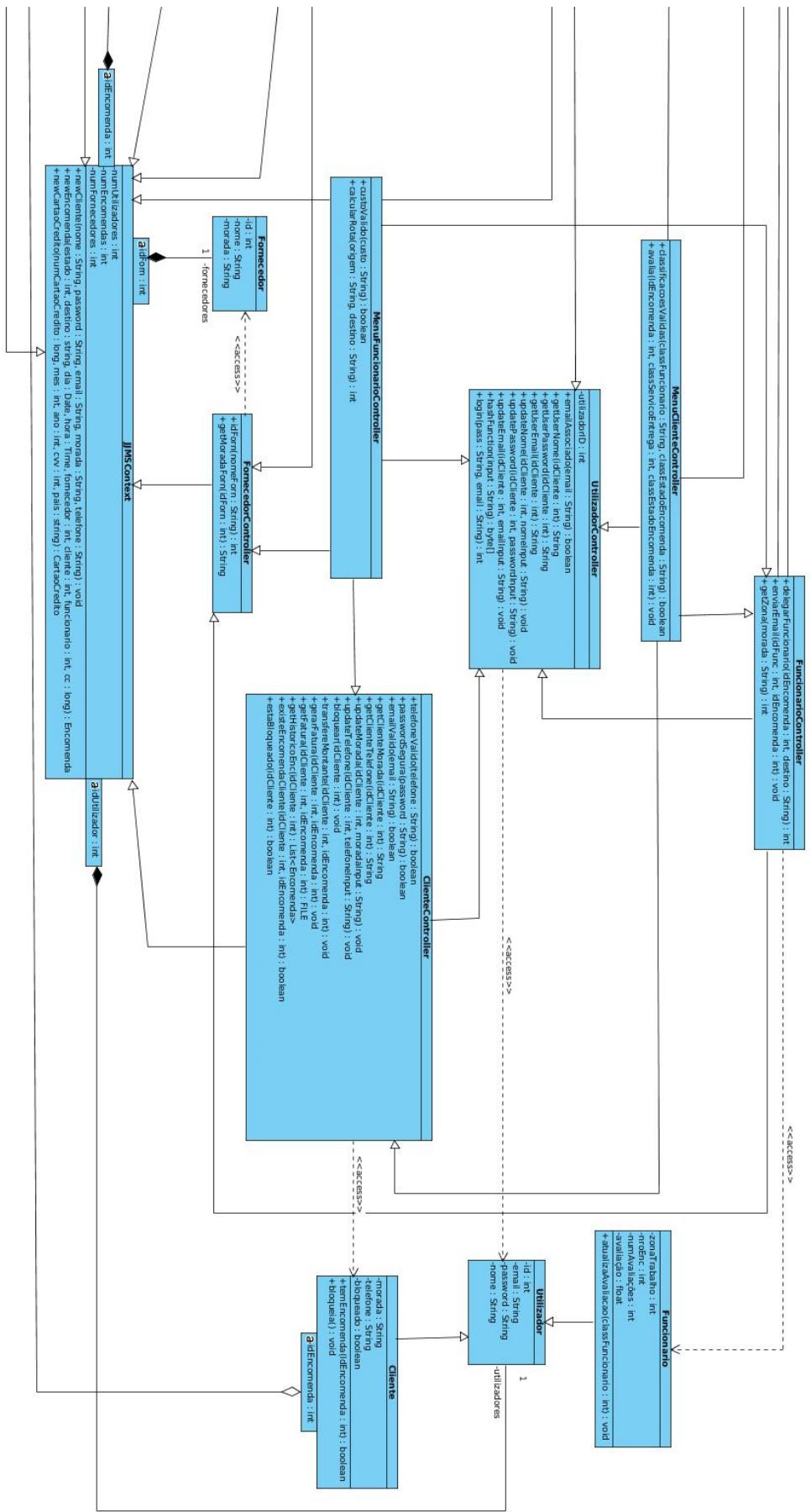
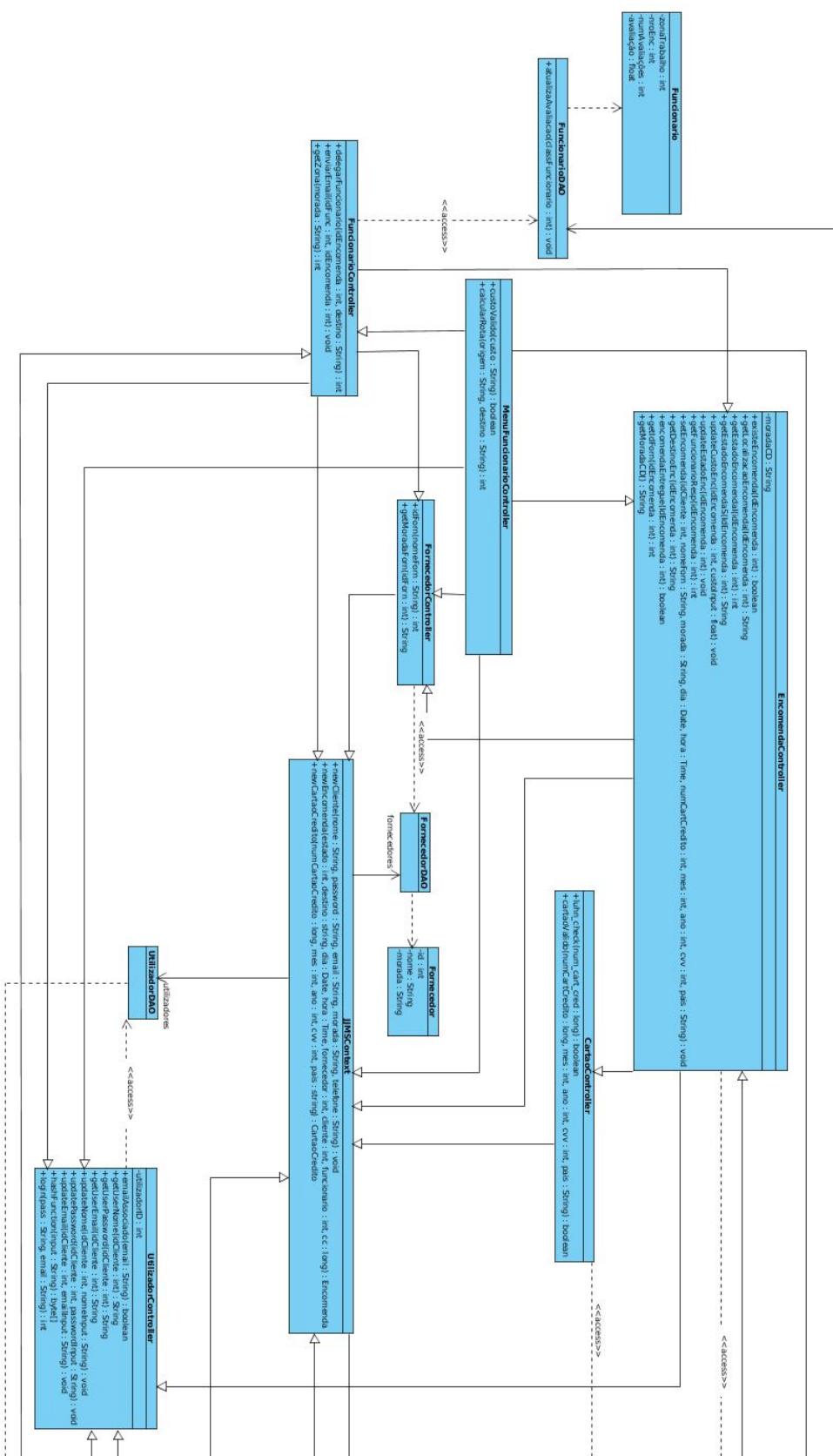


Figura 38 - Novo Diagrama de Classes sem persistência

Diagrama de Classes com persistência



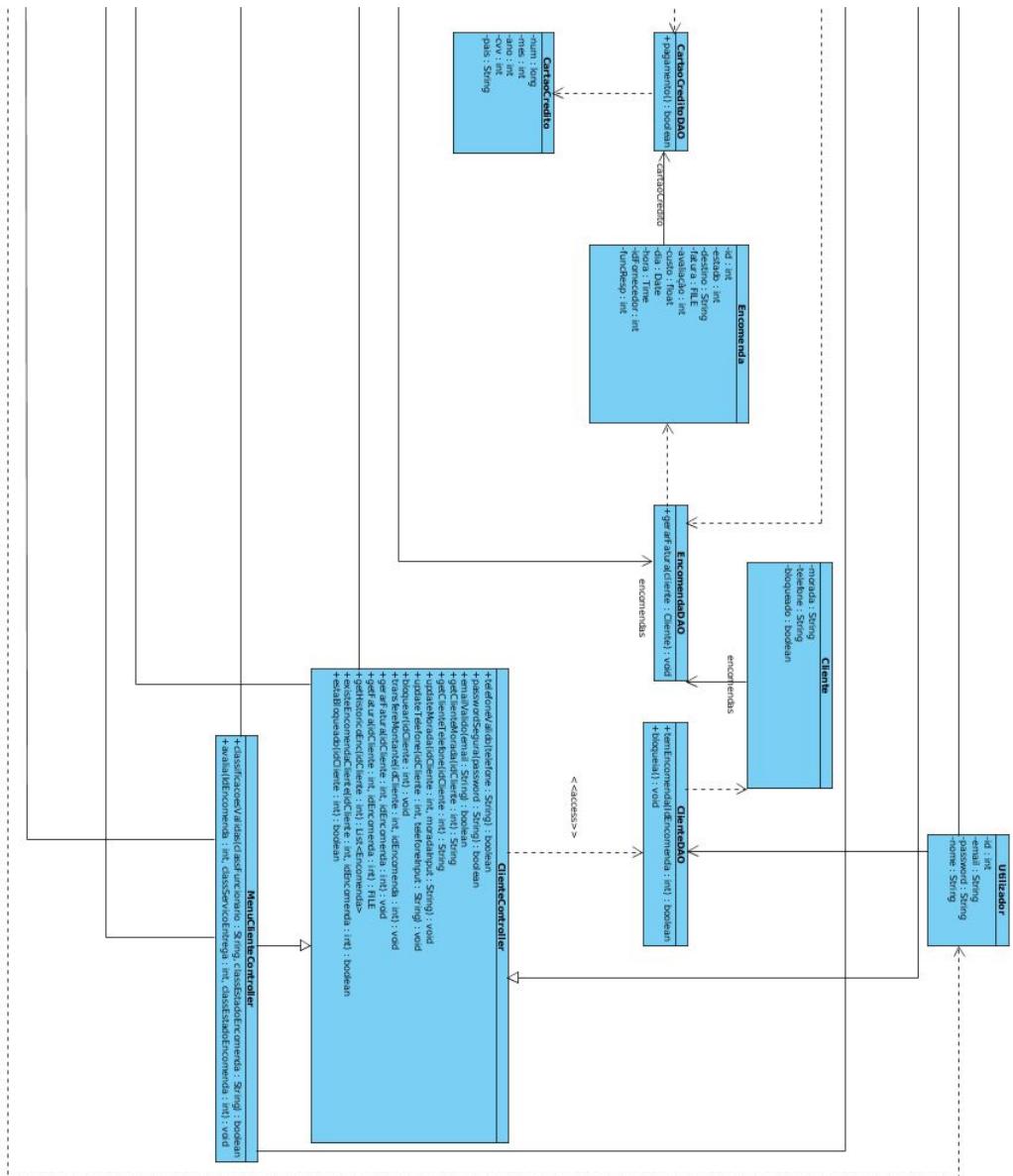


Figura 39 - Novo Diagrama de Classes com persistência

Já apresentado anteriormente foi o novo modelo lógico para a base de dados que foi gerado em consequência do uso da **EntityFramework**. Por fim as mudanças anteriormente referidas refletiram-se também nos diagramas de sequência e diagramas de sequência de sistema desenvolvidos e são apresentadas em anexo.

3.4. Funcionalidades efetivamente implementadas vs medidas de sucesso

Findo a fase de implementação é importante identificar as medidas de sucesso que foram atingidas e as que ficaram por implementar. Uma das medidas em causa consistia na conclusão do trabalho dentro do prazo previsto, algo não alcançado visto que ficou por

implementar a *interface Mobile* (tanto para cliente como para funcionário), para além de que a *interface Web* beneficiaria de uma melhoria no que diz respeito à *interface* com o utilizador de modo a torná-la mais intuitiva. Por outro lado não foram implementadas certas funções (getZona, CalcularRota, Pagamento, gerarFatura), que recorrem a API's externas (paypal API, google maps API).

Outra medida de sucesso proposta era um serviço fiável com tolerância de entrega de X minutos em que X depende da distância a percorrer, contudo esta medida de sucesso só pode ser comprovada com a integração do sistema na empresa ou através de estudos de mercado.

Em relação às medidas de segurança, nomeadamente o uso de algoritmos de hash para guardar informações como as credenciais dos utilizadores foram atingidas sendo as passwords dos utilizadores processadas pelo algoritmo SHA-384 escolhido por ter um bom compromisso entre comprimento da hash resultante e velocidade de hash.

As funcionalidades implementadas foram bem conseguidas e permitem ter uma ideia do quanto útil esta aplicação, após totalmente implementada, teria representando assim um sistema com grande valor/utilidade para a empresa/cliente em questão.

3.5. Instalação

Os passos para a instalação são:

1. Migração da base de dados para o SGBD do servidor hospedeiro(se a BD onde foi desenvolvida a app possuir dados a migrar)
2. Criação/Compilação da app para uma pasta do servidor hospedeiro (pode ser usado na linha de comandos dentro da pasta com o código o seguinte comando -> **dotnet publish -o <caminhoOndePublicar>**)
3. Configurar um gestor de processos que começa a app sempre que chega um pedido ou recomeça a app quando crasha ou o servidor reinicia
 - a. Gestores de Processos mais usados:
 - i. Linux
 1. Nginx
 2. Apache
 - ii. Microsoft
 1. Internet Information Services
 2. Windows Service
 4. Se desejado pode depois ser configurado um reverse proxy que encaminhe os pedidos para a app

4. Conclusões e Trabalho Futuro

Em relação à primeira fase do trabalho, com o objetivo de realizar uma fundamentação do trabalho a realizar, achamos que o projeto ficou bem definido e mais fácil de implementar, tendo em conta o rigor, a organização e os objetivos definidos.

Já no que diz respeito à segunda fase de desenvolvimento do projeto, com a ordem definida pelo grupo de trabalho e organização implementada, conseguimos eleger e definir com critério os requisitos do sistema a implementar, e caracterizar a fundo todos os use cases que conseguimos extrair dos mesmos, procedendo à realização de tabelas de especificação destes, desenho dos diagramas de sequência (divididos em 3 diferentes fases), desenho e definição de estados dentro da prototipagem/interface pretendida, construção do diagrama de classes com e sem DAO's (persistência), e, por fim, desenho da BD que servirá de suporte à aplicação.

Contudo, apesar de grande parte de especificação estar realizada, certos métodos tais como (pagamento, gerar, calcularRota, getZona, etc) que usam API's externas não ficaram especificadas em diagramas de sequência devido à sua complexidade, sendo assim, deixada a especificação das mesmas para o desenvolvimento.

Em relação à terceira e última fase, a implementação foi parcialmente feita, visto que apenas foi desenvolvido a página Web e a base de dados que a suporta, tendo contudo esta página Web ainda algumas falhas a corrigir(intuitividade da interface com o utilizador). No entanto o sistema permite a autenticação de utilizadores, registo e alteração de dados por parte dos clientes e requisição de encomendas bem como a consulta das encomendas previamente realizadas e a alteração do estado das mesmas por parte dos funcionários.

5. Referências Bibliográficas

- Chapter 4 - Requirements engineering. [online] Available at: <https://www.slideshare.net/software-engineering-book/ch4-req-eng> [Accessed 18 Mar. 2018].
- Thomas M. Connolly and Carolyn E. Begg, 2005. Database Systems: A Practical Approach to Design, Implementation, and Management. 4th ed. University of Paisley.
- José Creissac Campos / António Nestor Ribeiro, 2017. Modelação de comportamento / Máquinas de Estado. [online via BackBoard] Universidade do Minho Available at: <https://elearning.uminho.pt/bbcswebdav/pid-747063-dt-content-rid-1596177_1/courses/1718.H505N2_1/DSS-T09-M%C3%A1quinas%20de%20Estado.pdf> [Accessed Date 4 Abril 2018].
- José Creissac Campos / António Nestor Ribeiro, 2017. Diagramas de Sequência I. [online via BackBoard] Universidade do Minho Available at: <https://elearning.uminho.pt/bbcswebdav/pid-749131-dt-content-rid-1602415_1/courses/1718.H505N2_1/DSS-T11-Diag%20Seq%201.pdf> [Accessed Date 5 Abril 2018].
- José Creissac Campos / António Nestor Ribeiro, 2017. Modelação Comportamental / Diagramas de Sequência II. [online via BackBoard] Universidade do Minho Available at: <https://elearning.uminho.pt/bbcswebdav/pid-750757-dt-content-rid-1609727_1/courses/1718.H505N2_1/DSS-T15-Diag%20Seq%202.pdf> [Accessed Date 6 Abril 2018].
- José Creissac Campos / António Nestor Ribeiro, 2017. Modelação Estrutural / Diagramas de Classe II. [online via BackBoard] Universidade do Minho Available at: <https://elearning.uminho.pt/bbcswebdav/pid-753345-dt-content-rid-1621154_1/courses/1718.H505N2_1/DSS-T16-Diagramas%20de%20Classe%202.pdf> [Accessed Date 8 Abril 2018].
- José Creissac Campos / António Nestor Ribeiro, 2017. ORM (Object-Relational Mapping). [online via BackBoard] Universidade do Minho Available at: <https://elearning.uminho.pt/bbcswebdav/pid-759417-dt-content-rid-1646212_1/courses/1718.H505N2_1/DSS-T22T23-ORM.pdf> [Accessed Date 9 Abril 2018].
- Inheritance with EF Code First: Part 1 – Table per Hierarchy (TPH). [online] Available at:

<https://weblogs.asp.net/manavi/inheritance-mapping-strategies-with-entity-framework-code-first-ctp5-part-1-table-per-hierarchy-tph> [Accessed 9 Jun. 2018].

- C# Reference. [online] Available at: <https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/index> [Accessed 4 May 2018].
- Entity Framework Core Quick Overview. [online] Available at: <https://docs.microsoft.com/en-us/ef/core/> [Accessed 8 May 2018].
- Introduction to ASP.NET Core. [online] Available at: <https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-2.1> [Accessed 5 May 2018].
- .NET Core Guide. [online] Available at: <https://docs.microsoft.com/en-us/dotnet/core/> [Accessed 7 May 2018].

Lista de Siglas e Acrónimos

JJMS	João José Miguel Simão
SQL	Structured Query Language
SGBD	Sistema de Gestão de Base de Dados
UML	Unified Modeling Language
BD	Base de Dados
ER	Entidade-Relacionamento
DAO	Data Access Object
MVC	Model-View-Controller
QA	Quality Assurance
TPH	Table-Per-Hierarchy
HTTP	Hypertext Transfer Protocol
API	Application Programming Interface

Anexos

Especificação de Use Cases

Use Case: Autenticar

Pré-condição: Cliente/Funcionário está na página/aplicação da JJMS

Pós-condição: Autenticação bem sucedida

	Cliente	Sistema
Comportamento Normal	1. Indica que pretende fazer login	
		2. Apresenta campos de dados de login (password, email)
	3. Introduz a password e email	
		4. Verifica dados
		5. Informa que o login foi realizado com sucesso
Exceção 1 [email inexistente] (passo 4)		4.1.1. Informa que email inserido não existe
		4.1.2. Termina sem sucesso
Exceção 2 [password errada] (passo 4)		4.1.1. Informa que a password está errada
		4.1.2. Termina sem sucesso

Use Case: Consultar Histórico

Pré-condição: Cliente estar autenticado

Pós-condição: Consulta com sucesso do histórico de encomendas

	Cliente	Sistema
Comportamento Normal	1. Indica que pretende	

	consultar o histórico de encomendas	
		2. Apresenta encomendas já entregues ao cliente, com o respectivo código
		3. Termina operação
Comportamento Alternativo 1 [o cliente pretende consultar a fatura de uma encomenda] (passo 3)	3.1. Seleciona encomenda	
		3.2. Apresenta fatura da encomenda

Use Case: Tracking Encomenda

Pré-condição: Cliente estar autenticado

Pós-condição: Consulta com sucesso localização e estado da encomenda

	Cliente	Sistema
Comportamento Normal	1. Indica que pretende consultar a localização e estado atual da encomenda	
		2. Pede código da encomenda em causa
	3. Insere código	
		4. Mostra localização e estado atual da encomenda
Exceção 1 [Cliente cancela operação] (passo 3)	3.1. Cancela operação	
		3.2. Termina sem sucesso
Exceção 2 [Código de encomenda não existe] (passo 4)		4.1. Informa que não existe encomenda com o código inserido
		4.2. Termina sem sucesso

Use Case: Avaliar Serviço

Pré-condição: Cliente estar autenticado

Pós-condição: Avalia serviço relacionado com uma dada encomenda

	Cliente	Sistema
Comportamento Normal	1. Informa que pretende efetuar avaliação do serviço prestado na entrega de uma	

	encomenda	
		2. Pede código da encomenda
	3. Insere código da encomenda	
		4. Verifica se a encomenda existe e, em caso afirmativo, se a mesma já foi entregue
		5. Pede uma classificação no intervalo [0-10] para o serviço prestado pelo funcionário que realizou a entrega e outra no intervalo [0-5] para o estado da encomenda
	6. Insere classificações	
		7. Verificar classificações inseridas
		8. Informa que a avaliação foi feita com sucesso
Exceção 1 [Cliente cancela operação] (passo 3)	3.1. Cancela a operação	
		3.2. Termina sem sucesso
Exceção 2 [Código de encomenda não existe] (passo 4)		4.1.1. Informa que não existe encomenda com o código inserido
		4.1.2. Termina sem sucesso
Exceção 3 [Encomenda ainda não foi entregue] (passo 4)		4.2.1. Informa que a encomenda em causa ainda não foi entregue e, por isso, a avaliação ainda não pode ser realizada
		4.2.2. Termina sem sucesso
Exceção 4 [Cliente cancela operação] (passo 6)	6.1. Cancela a operação	
		5.2. Termina sem sucesso
Exceção 5 [Classificações inseridas não respeitam a gama de valores [0-10] para o serviço e [0-5] para o estado da encomenda] (passo 7)		7.1. Informa que foram inseridos valores incorretos, ou seja, não respeitando a gama de valores.
		7.2. Termina sem sucesso

Use Case: Registar

Pré-condição: Cliente está na página / aplicação da JJMS

Pós-condição: Registo como cliente na empresa efetuado com sucesso

	Cliente	Sistema
Comportamento Normal	1. Indica que pretende efetuar registo como cliente na JJMS	
		2. Pede dados necessários para o registo (nome, password, email, morada, telefone)
	3. Insere os dados pedidos	
		4. Verifica dados inseridos
		5. Informa que realizou registo com sucesso
Exceção 1 [Cliente cancela operação] (passo 3)	3.1. Cancela a operação	
		3.2 Termina sem sucesso
Exceção 2 [Campos por preencher] (passo 4)		4.1.1. Informa que não foram preenchidos todos os campos pedidos
		4.1.2. Termina sem sucesso
Exceção 3 [Email já associado a outro cliente] (passo 4)		4.2.1. Informa que email inserido já está associado a outro cliente
		4.2.2. Termina sem sucesso
Exceção 4 [Telefone não é um número] (passo 4)		4.3.1. Informa que telefone inserido não é válido
		4.3.2. Termina sem sucesso
Exceção 5 [Password não respeita os níveis mínimos de segurança] (passo 4)		4.4.1. Informa que password não possui os requisitos mínimos de segurança (com 8 ou mais caracteres e possuir números, letras e símbolos)
		4.4.2. Termina sem sucesso

Use Case: Alterar Dados

Pré-condição: Cliente estar autenticado

Pós-condição: Valores alterados com sucesso

	Cliente	Sistema
Comportamento Normal	1. Indica que pretende alterar os seus dados pessoais	
		2. Apresenta dados atuais e passíveis de alterar (nome, password, email, morada, telefone)
	3. Altera os dados	
		4. Verifica dados alterados
		5. Atualiza dados
		6. Informa que os dados foram alterados com sucesso
Exceção 1 [email já associado a outro cliente] (passo 4)		4.1.1. Informa que email inserido já está associado a outro cliente
		4.1.2. Termina sem sucesso
Exceção 2 [telefone não é um número] (passo 4)		4.2.1. Informa que telefone inserido não é válido
		4.2.2. Termina sem sucesso
Exceção 3 [password não respeita os níveis mínimos de segurança, isto é, com 8 ou mais caracteres e possuir letras, números e símbolos] (passo 4)		4.3.1. Informa que password não possui os requisitos mínimos de segurança (com 8 ou mais caracteres e possuir números, letras e símbolos)
		4.3.2. Termina sem sucesso

Use Case: Atualizar Estado

Pré-condição: Funcionário estar autenticado e possuir uma encomenda

Pós-condição: Estado atualizado com sucesso

	Funcionário	Sistema
Comportamento Normal	1. Indica que pretende alterar estado da encomenda que possui	
		2. Pergunta o código da encomenda

	3. Insere o código da encomenda que pretende alterar	
		4. Verifica se o código pertence a uma encomenda, e se é uma encomenda válida, ou seja que ainda não foi entregue.
		5. Pede o custo do transporte
	6. Insere o custo do transporte	
		7. Verifica se o custo é válido.
		8. Adicionar o custo fornecido ao já presente na encomenda
		9. Altera o estado da encomenda (“no centro de distribuição” para “em trânsito”)
		10. Informa que o estado foi atualizado com sucesso
Exceção 1 [Código de encomenda não existe, ou encomenda já foi entregue] (passo 4)		4.1. Informa que não existe encomenda com esse código, ou que a encomenda já foi entregue.
		4.2. Termina sem sucesso.
Exceção 2 [Valor de custo inválido, possui letras ou símbolos, ou é negativo] (passo 7)		7.1. Informa que custo inserido é inválido visto que ou possui letras ou símbolos, ou que o custo é negativo.
		7.2. Termina sem sucesso.
Comportamento Alternativo 1 [Estado passa de “com o fornecedor” para “no centro de distribuição”] (passo 9)		9.1.1. Delegar funcionário para realizar entrega ao cliente (<<include>> Delegar Funcionários)
		9.1.2. Regressar a 10.
Comportamento Alternativo 2 [Estado passa de “em trânsito” para “entregue”] (passo 9)		9.2.1. Realizar pagamento do serviço. (<<include>> Pagar Serviço).
		9.2.2 Regressar a 10.

Use Case: Pagar Serviço

Pré-condição: possuir dados de pagamento do cliente

Pós-condição: pagamento efetuado com sucesso

	Sistema
Comportamento Normal	<ol style="list-style-type: none"> 1. Transferir montante correspondente ao custo do transporte da encomenda entregue. 2. Gerar fatura correspondente ao pagamento.
Exceção 1 [Transferência falhou] (passo 1)	<ol style="list-style-type: none"> 1.1. Bloquear a conta do cliente, ou seja, impedir que faça novos requisitos de encomendas 1.2. Termina sem sucesso

Use Case: Consultar Rota

Pré-condição: Funcionário estar autenticado e ser-lhe delegado uma encomenda

Pós-condição: Funcionário consulta rota com sucesso

	Funcionário	Sistema
Comportamento Normal	<ol style="list-style-type: none"> 1. Pede para consultar rota 	
		<ol style="list-style-type: none"> 2. Pede código da encomenda e localização do funcionário
	<ol style="list-style-type: none"> 3. Insere código e localização 	
		<ol style="list-style-type: none"> 4. Verifica se existe encomenda com esse código e se não foi já entregue
		<ol style="list-style-type: none"> 5. Verifica funcionário encarregue da encomenda
		<ol style="list-style-type: none"> 6. Calcula rota(<<include>> calcular rota)
		<ol style="list-style-type: none"> 7. Apresenta rota
Exceção 1 [encomenda não existe ou foi já entregue] (passo 4)		<ol style="list-style-type: none"> 4.1. Informa que encomenda não existe ou já foi entregue
		<ol style="list-style-type: none"> 4.2. Termina sem sucesso
Exceção 2 [encomenda ainda não foi delegada ou foi delegada a outro funcionário] (passo 5)		<ol style="list-style-type: none"> 5.1. Informa que a encomenda não lhe foi delegada
		<ol style="list-style-type: none"> 5.2. Termina sem sucesso

Use Case: Delegar Funcionários

Pré-condição: Encomenda não entregue

Pós-condição: Encomenda delegada a funcionário

	Sistema
Comportamento Normal	<ol style="list-style-type: none">1. Obter estado da encomenda2. Verifica que o estado é “no fornecedor”3. Obter morada do fornecedor4. Aplicar algoritmo de modo a escolher o funcionário a transportar a encomenda tendo em conta a morada do fornecedor5. Delegar a encomenda ao funcionário escolhido e enviar um email ao mesmo com os dados da encomenda(código, origem, destino)
Comportamento Alternativo 1 [Estado da encomenda é “no centro de distribuição”] (passo 2)	<ol style="list-style-type: none">2.1. Obter morada de entrega2.2. Aplicar algoritmo de modo a escolher o funcionário a transportar a encomenda tendo em conta a morada do cliente
	<ol style="list-style-type: none">2.3. Regressar a 4

Use Case: Calcular Rota

Pré-condição: Possuir código da encomenda e localização do funcionário

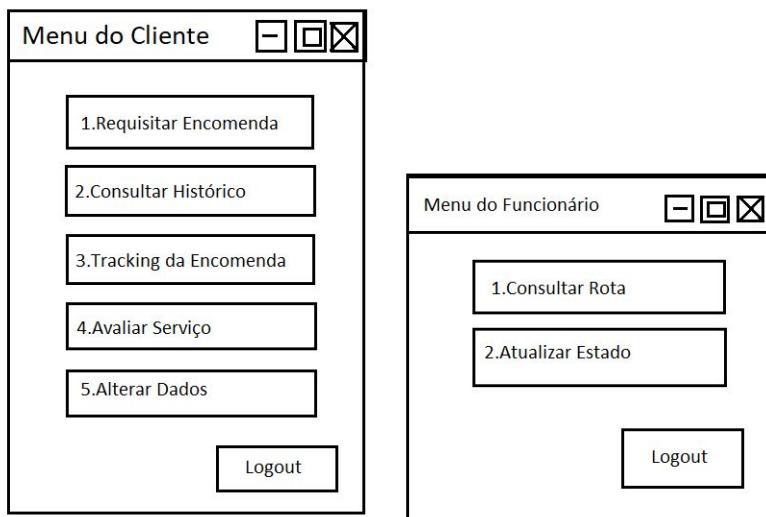
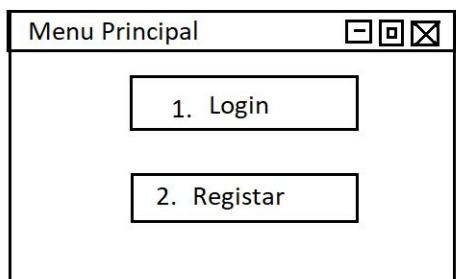
Pós-condição: Rota calculada com sucesso

	Sistema
Comportamento Normal	<ol style="list-style-type: none">1. Obtém estado da encomenda2. Verifica que o estado da encomenda é “em trânsito”3. Obtém morada de entrega da encomenda4. Obtém rota desde a localização do funcionário até à morada de entrega através de um programa terceiro (google maps, via michelin)
Comportamento Alternativo 1 [Encomenda possui o estado “com o fornecedor”] (passo 2)	<ol style="list-style-type: none">2.1.1. Obtém morada do fornecedor2.1.2. Obtém rota desde a localização do funcionário até à morada do fornecedor através de um programa terceiro (google maps, via michelin)

Comportamento Alternativo 2 [Encomenda possui o estado “no centro de distribuição”] (passo 2)	2.2.1. Obtém morada do centro de distribuição onde está a encomenda 2.2.2. Obtém rota desde a localização do funcionário até à morada do centro de distribuição através de um programa terceiro (google maps, via michelin)
--	--

Prototipagem dos Use Cases

Geral



Autenticar

Autenticar	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Email:	<input type="text"/>		
Senha:	<input type="password"/>		
OK			

Email inexistente	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
O email inserido não existe.			
Voltar			

Senha inválida	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
A Senha inserida é inválida.			
Voltar			

Consultar Histórico

Consultar Encomendas	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>						
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td>Código 1</td> <td>Encomenda 1</td> </tr> <tr> <td>Código 2</td> <td>Encomenda 2</td> </tr> <tr> <td>Código 3</td> <td>Encomenda 3</td> </tr> </table>				Código 1	Encomenda 1	Código 2	Encomenda 2	Código 3	Encomenda 3
Código 1	Encomenda 1								
Código 2	Encomenda 2								
Código 3	Encomenda 3								
Voltar									

Fatura da Encomenda	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Código : Código 1 Encomenda : Encomenda 1 Fatura : Fatura 1			
Voltar			

Tracking Encomenda

Tracking Encomenda	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Insira o código da sua encomenda.			
<input type="text"/>			
Cancelar		Continuar	

Informação Encomenda	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Encomenda X31ZTY4Z <ul style="list-style-type: none"> • Localização : Lisboa • Estado atual : no centro de distribuição 			
OK			

Código inexistente	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Não existe a encomenda com o código inserido.			
OK			

Cancelar	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Operação cancelada.			
OK			

Avaliar Serviço

<p>Avalia serviço</p> <p>Insira o código da sua encomenda:</p> <input type="text"/> <p>Cancelar Continuar</p>	<p>Cancelar</p> <p>Operação cancelada.</p> <p>OK</p>
<p>Inserir classificações</p> <p>Avalie tendo em conta os seguintes critérios:</p> <ul style="list-style-type: none"> • Serviço prestado pelo funcionário [0-10] na entrega • Estado da encomenda [0-5] <p>Cancelar Avaliar</p>	<p>Código inexistente</p> <p>Não existe a encomenda com o código inserido.</p> <p>OK</p>
<p>Sucesso</p> <p>Avaliação efetuada com sucesso.</p> <p>OK</p>	<p>Encomenda por entregar</p> <p>A encomenda ainda não foi entregue, sendo assim, ainda não pode ser avaliada.</p> <p>OK</p>
	<p>Classificações inválidas</p> <p>Foram inseridos valores incorrectos, não respeitando a gama de valores estabelecida.</p> <p>OK</p>

Registrar

Register

Insira os dados necessários:

Nome: [Text Input]

Password: [Text Input]

Email: [Text Input]

Morada: [Text Input]

Telefone: [Text Input]

Cancelar

Operação cancelada.

Email em uso

O email inserido já se encontra associado a outro cliente.

Sucesso

Registo efetuado com sucesso.

Campos por preencher

Não preencheu todos os campos pedidos.

Telefone inválido

O Telefone inserido não é válido.

Password insuficiente

A Password não cumpre requisitos mínimos de segurança:

- 8 ou mais caracteres;
- possuir números, letras e símbolos.

Alterar Dados

<p>Alterar Dados :</p> <p>Nome : João</p> <p>Password : ****</p> <p>Email : d@gmail.com</p> <p>Morada : rua das vidas</p> <p>Telefone : 923345729</p> <p><input type="button" value="Cancelar"/> <input type="button" value="Alterar"/></p>	<p>Alterado com sucesso</p> <p>Dados alterados com sucesso.</p> <p><input type="button" value="OK"/></p> <p>Email já associado</p> <p>Email já associado a outro cliente.</p> <p><input type="button" value="OK"/></p> <p>Telefone inválido</p> <p>Telefone inserido não é válido</p> <p><input type="button" value="OK"/></p> <p>Password insuficiente</p> <p>Password não cumpre requisitos mínimos de segurança:</p> <ul style="list-style-type: none">• 8 ou mais caracteres• possui números, letras e símbolos. <p><input type="button" value="OK"/></p>
---	--

Atualizar Estado

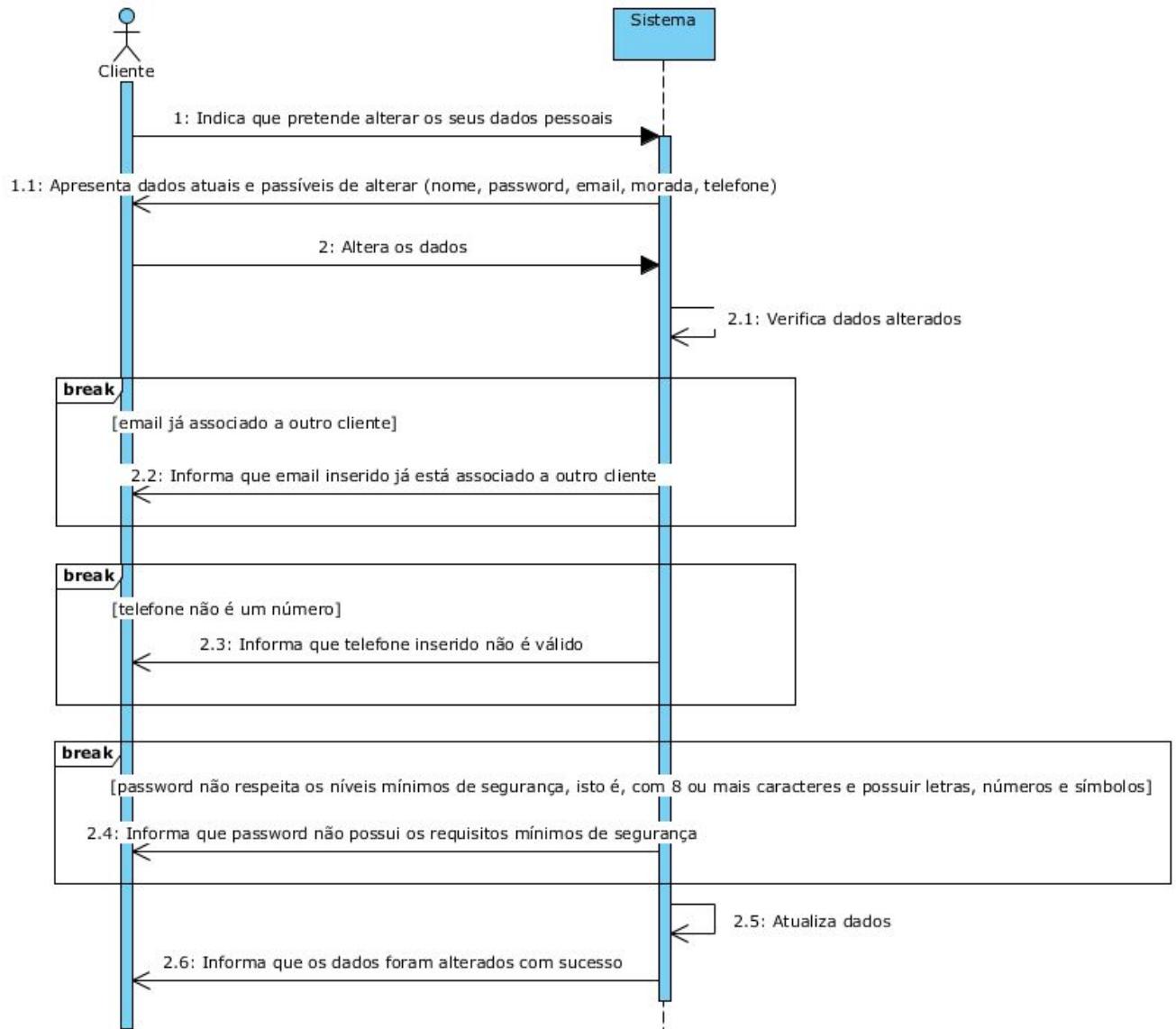
Código da Encomenda Código da encomenda: <input type="text"/>	Custo Transporte Custo do Transporte: <input type="text"/>	Atualizado com sucesso. Estado atualizado com sucesso. OK
		Não existe encomenda Não existe encomenda com o código fornecido ou já foi entregue. OK
Custo Inválido Custo inserido inválido, é decimal negativo ou possui letras ou símbolos. OK		

Consultar Rota

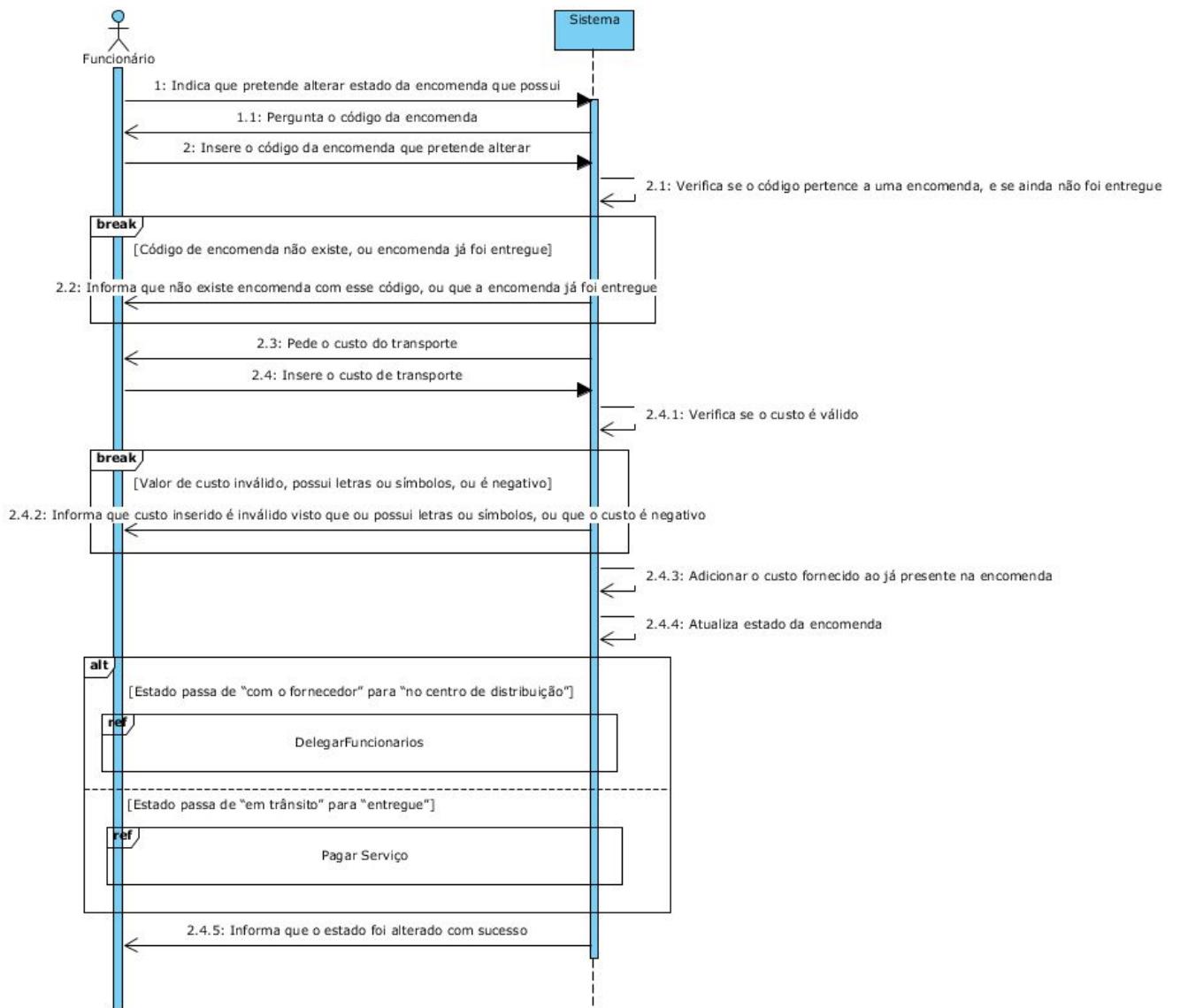
Consultar rota - <input type="checkbox"/> X	
Id Encomenda: <input type="text"/>	
Localização atual: <input type="text"/>	
<input type="button" value="Consultar"/>	
Encomenda Inexistente - <input type="checkbox"/> X	
Encomenda não existe <input type="checkbox"/> OK	
Encomenda Inválida - <input type="checkbox"/> X	
Encomenda delegada a outro fumeiomário. <input type="checkbox"/> OK	

**Diagramas de Sequência dos Use Cases (Uses Cases
-> Diagramas de Sequência)**

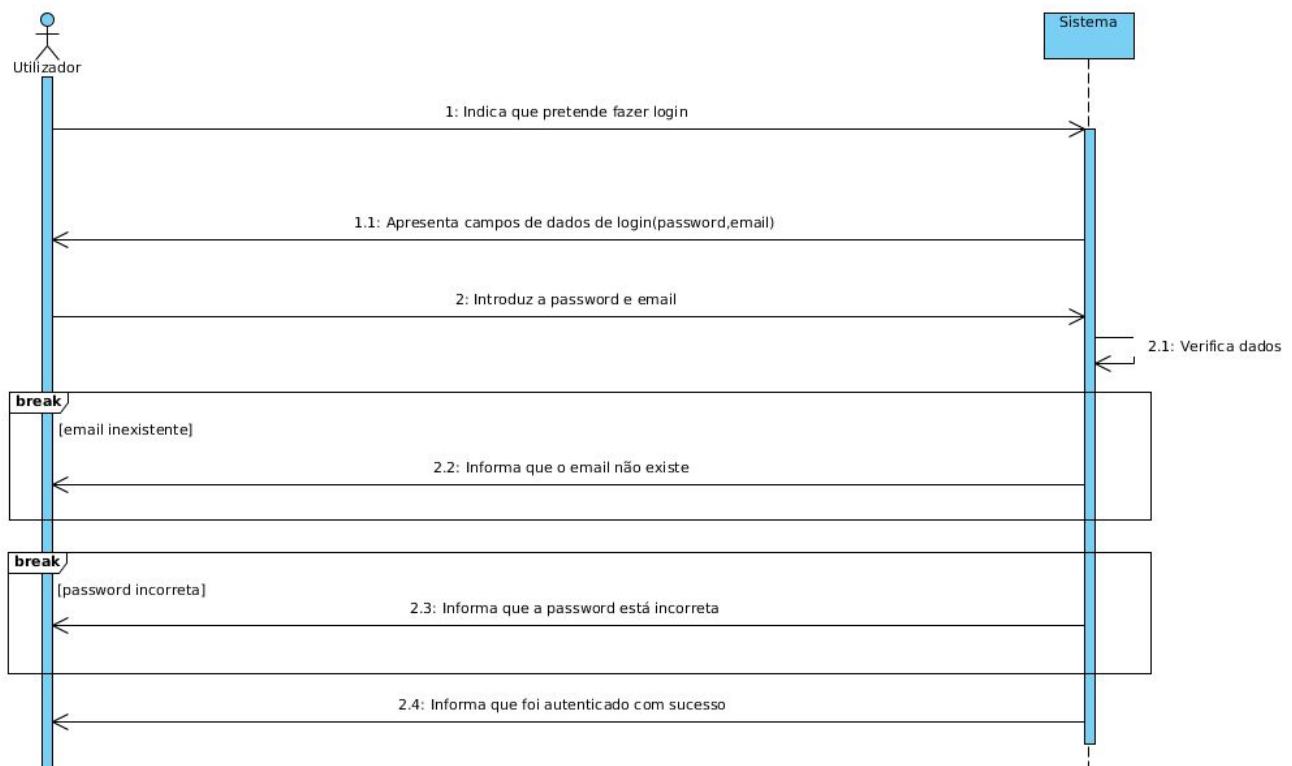
sd Alterar Dados



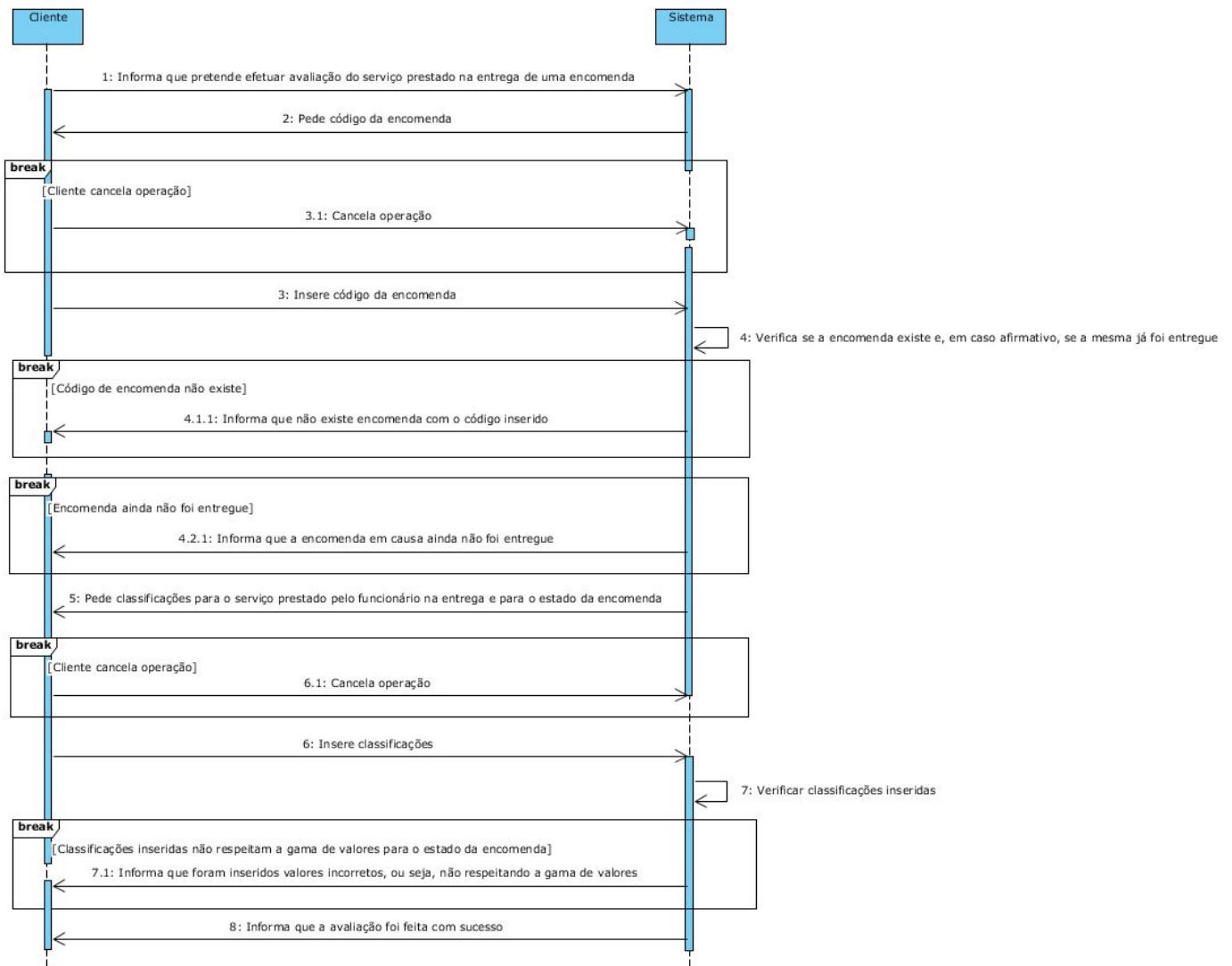
sd Atualizar Estado



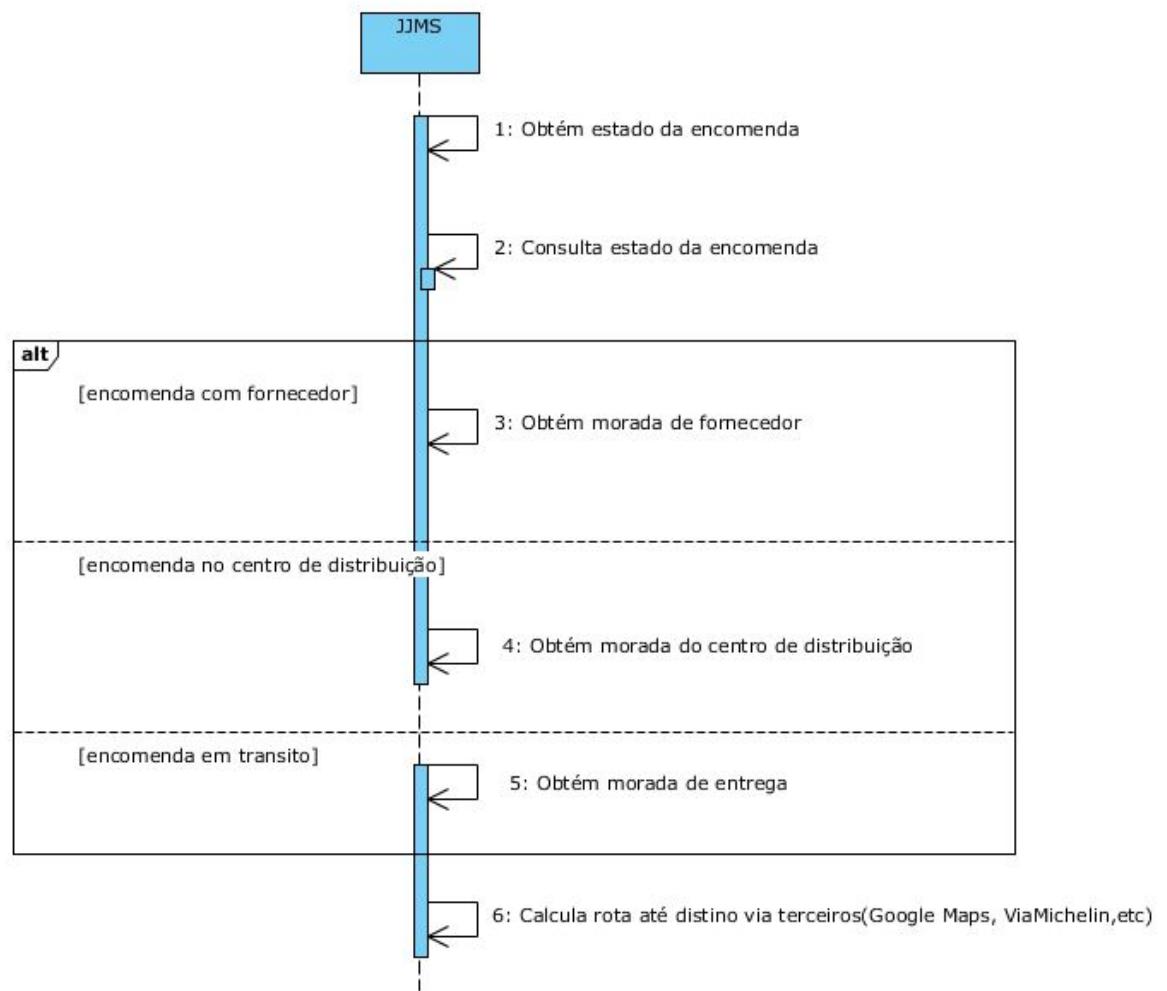
sd Autenticar



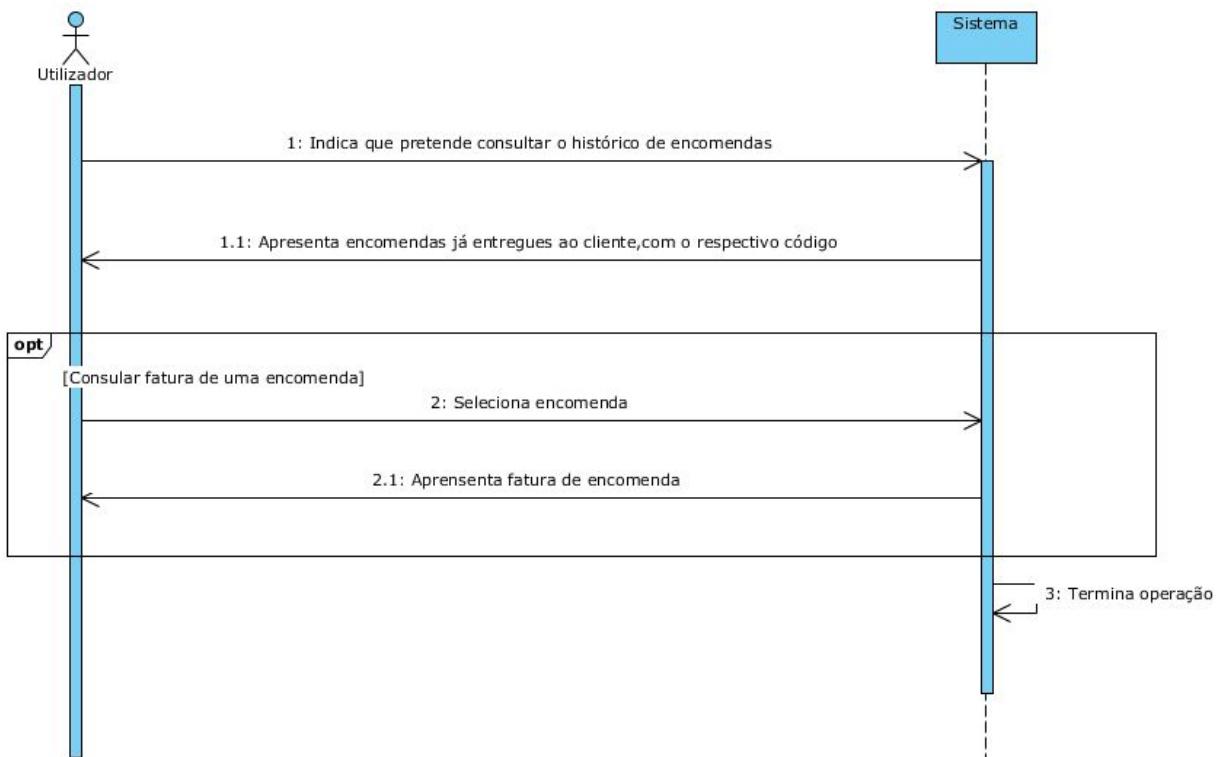
sd AvaliarServiço



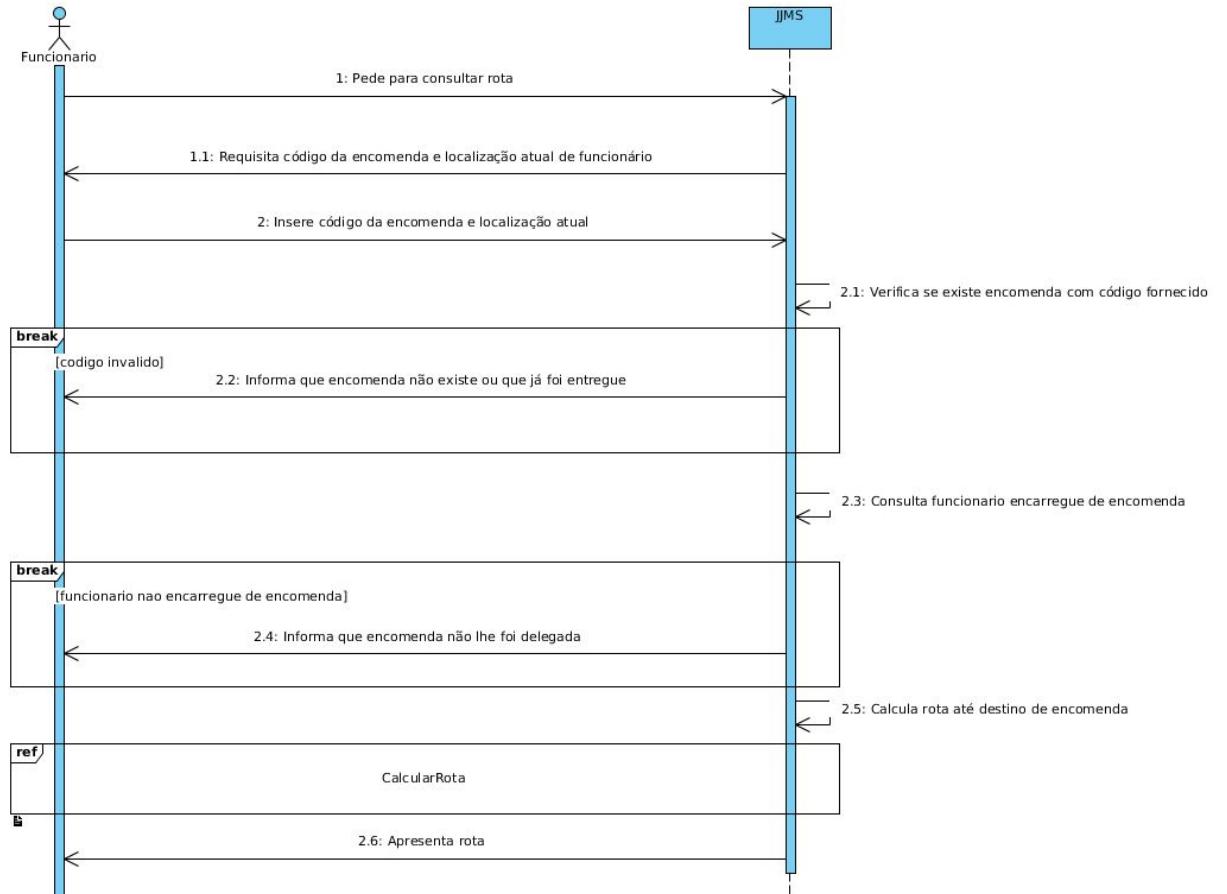
sd CalcularRota



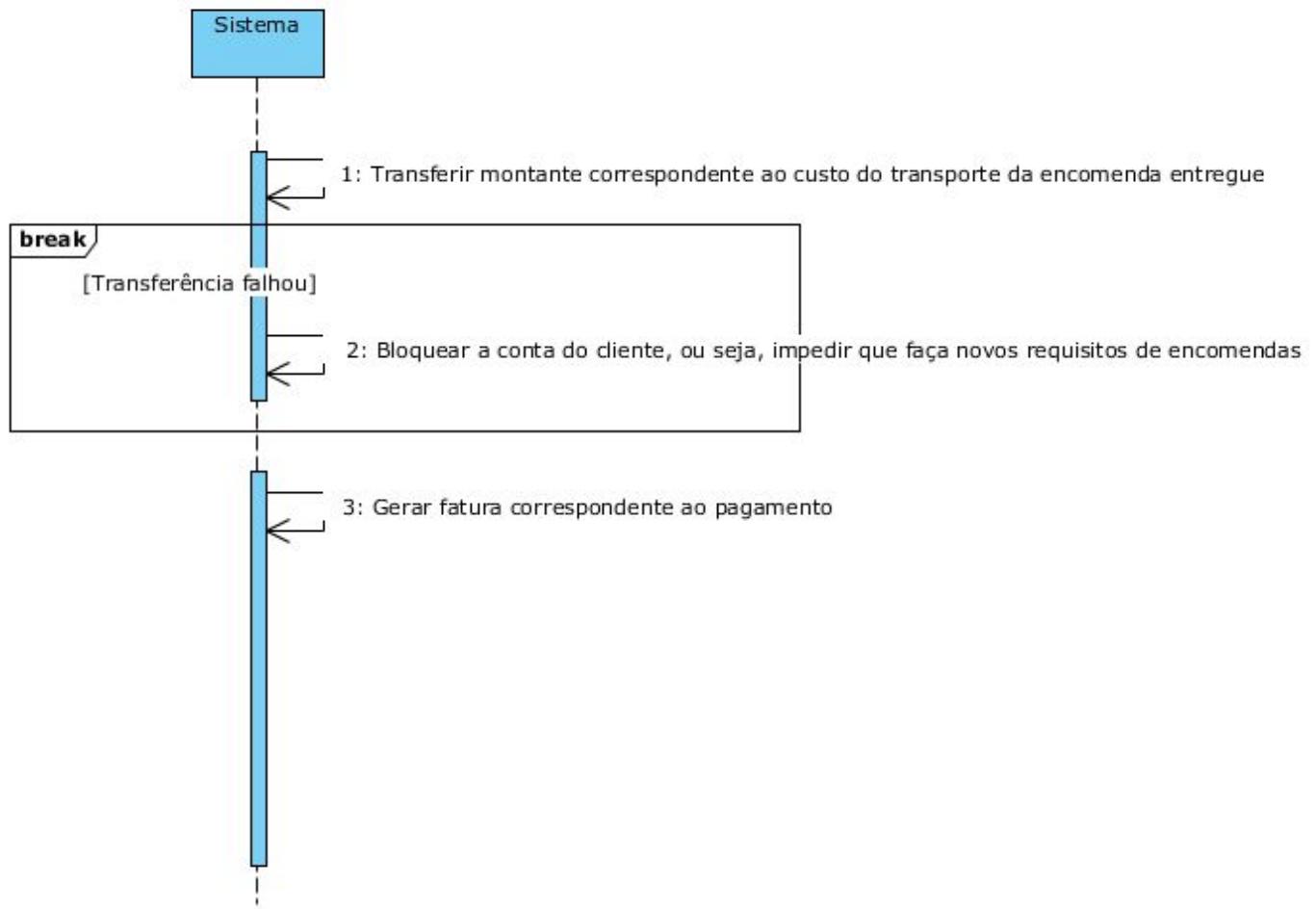
sd Consultar histórico



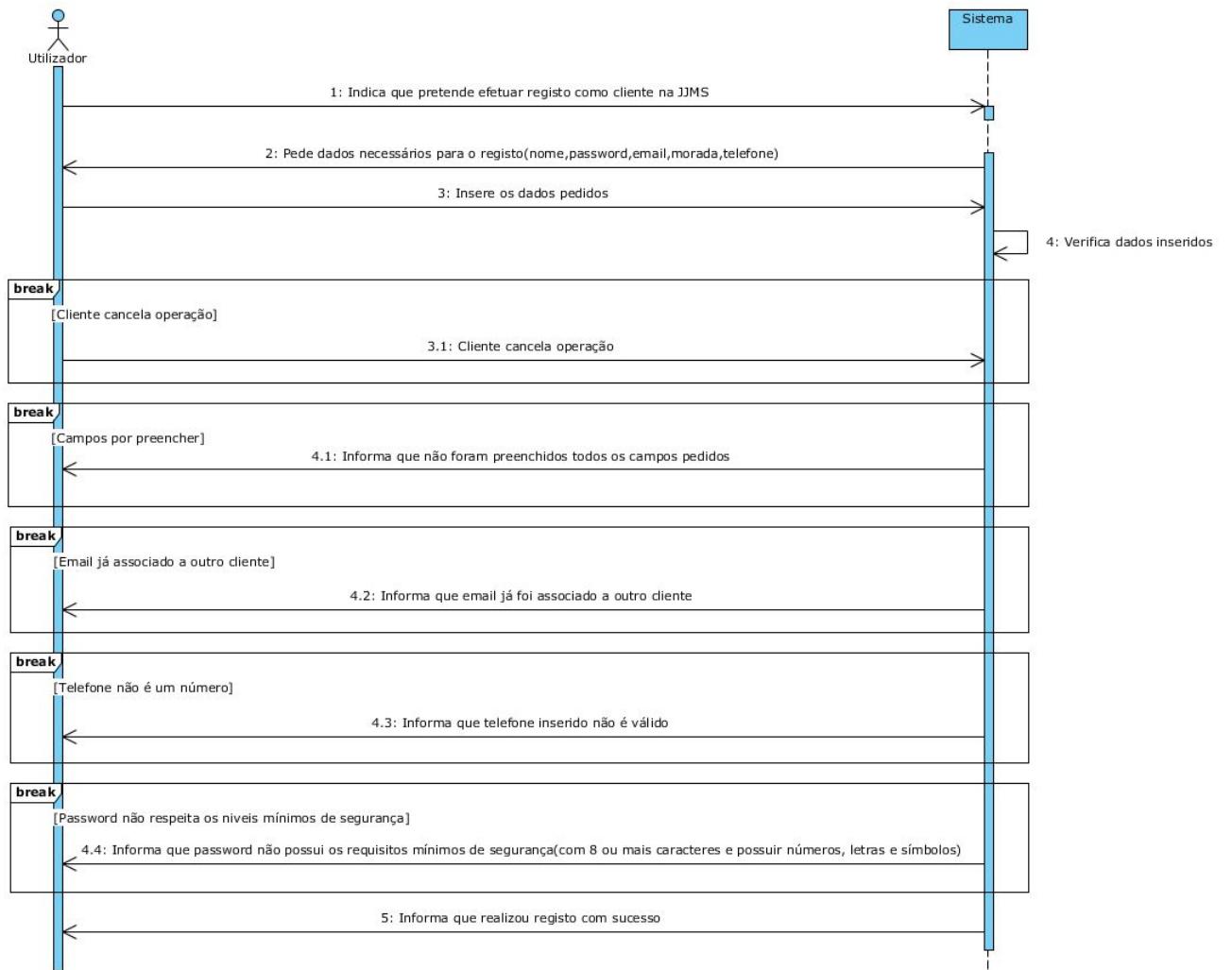
sd ConsultarRota



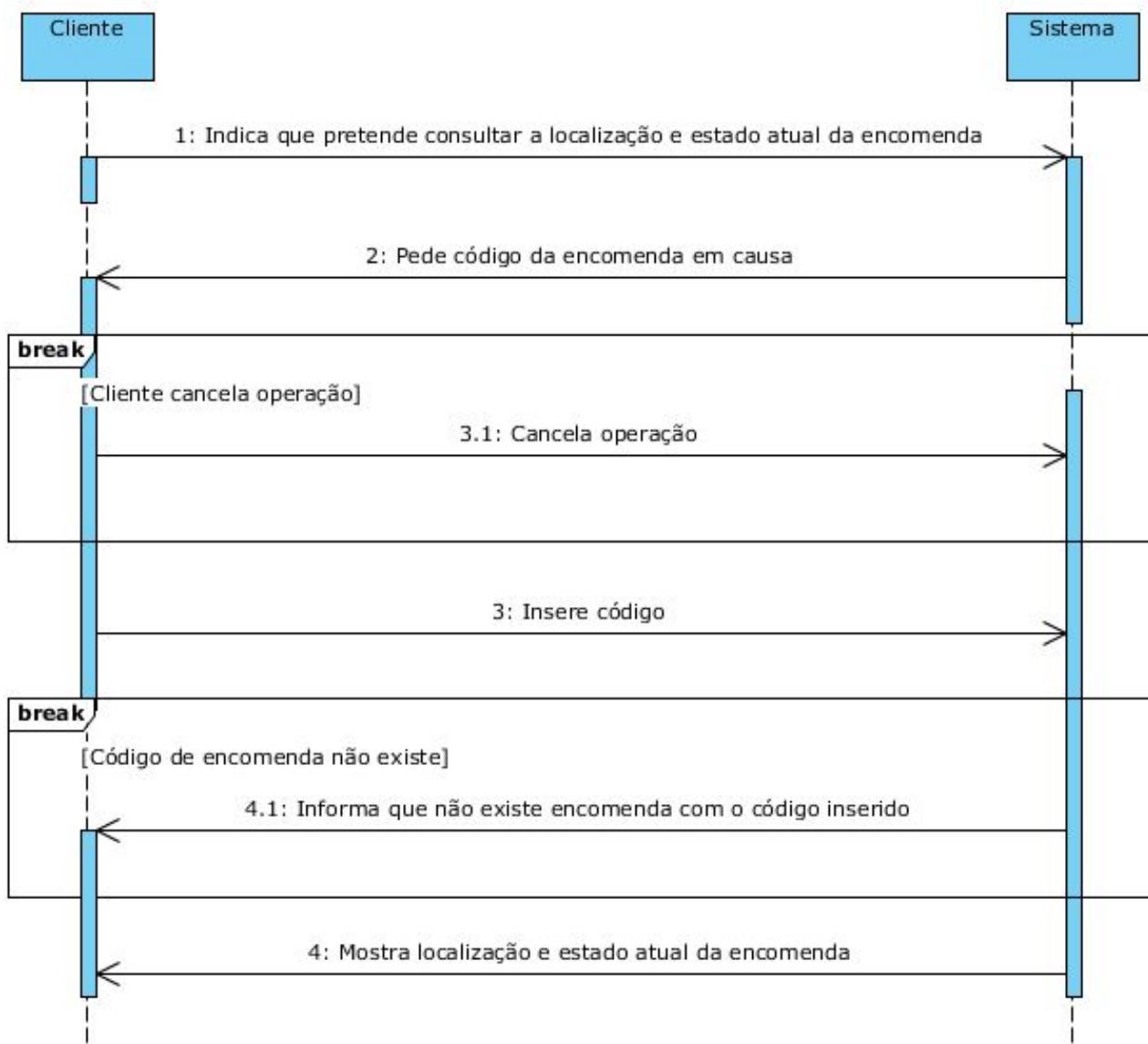
sd Pagar Serviço



sd Registrar

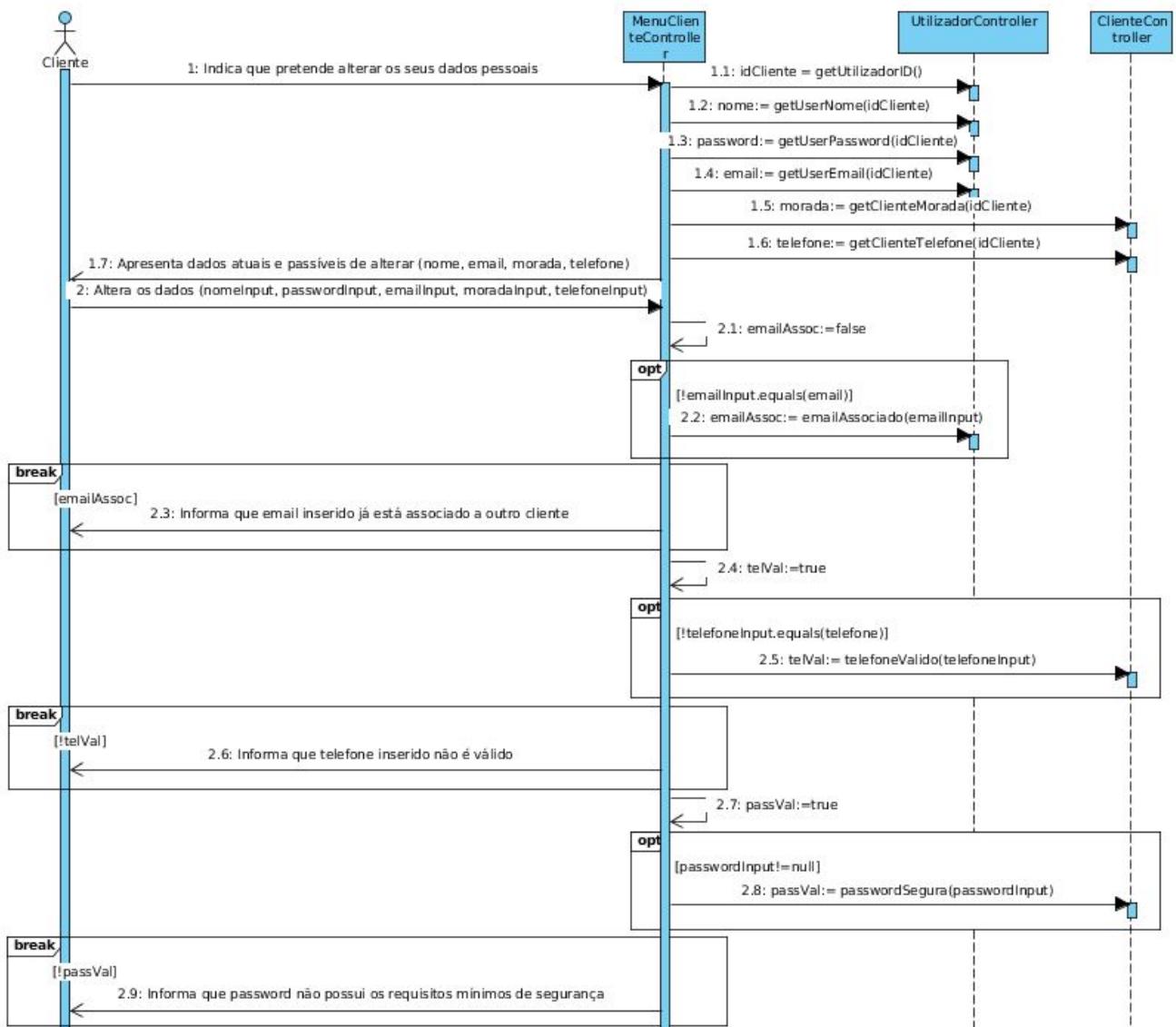


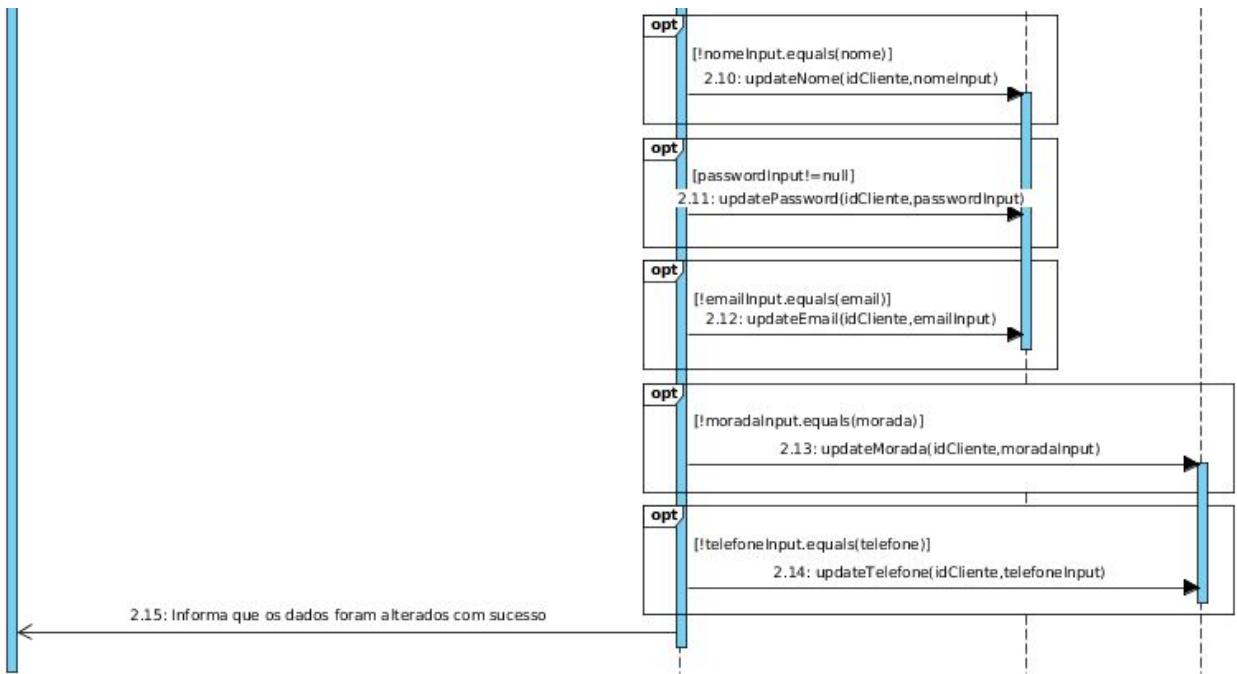
sd TrackingEncomenda



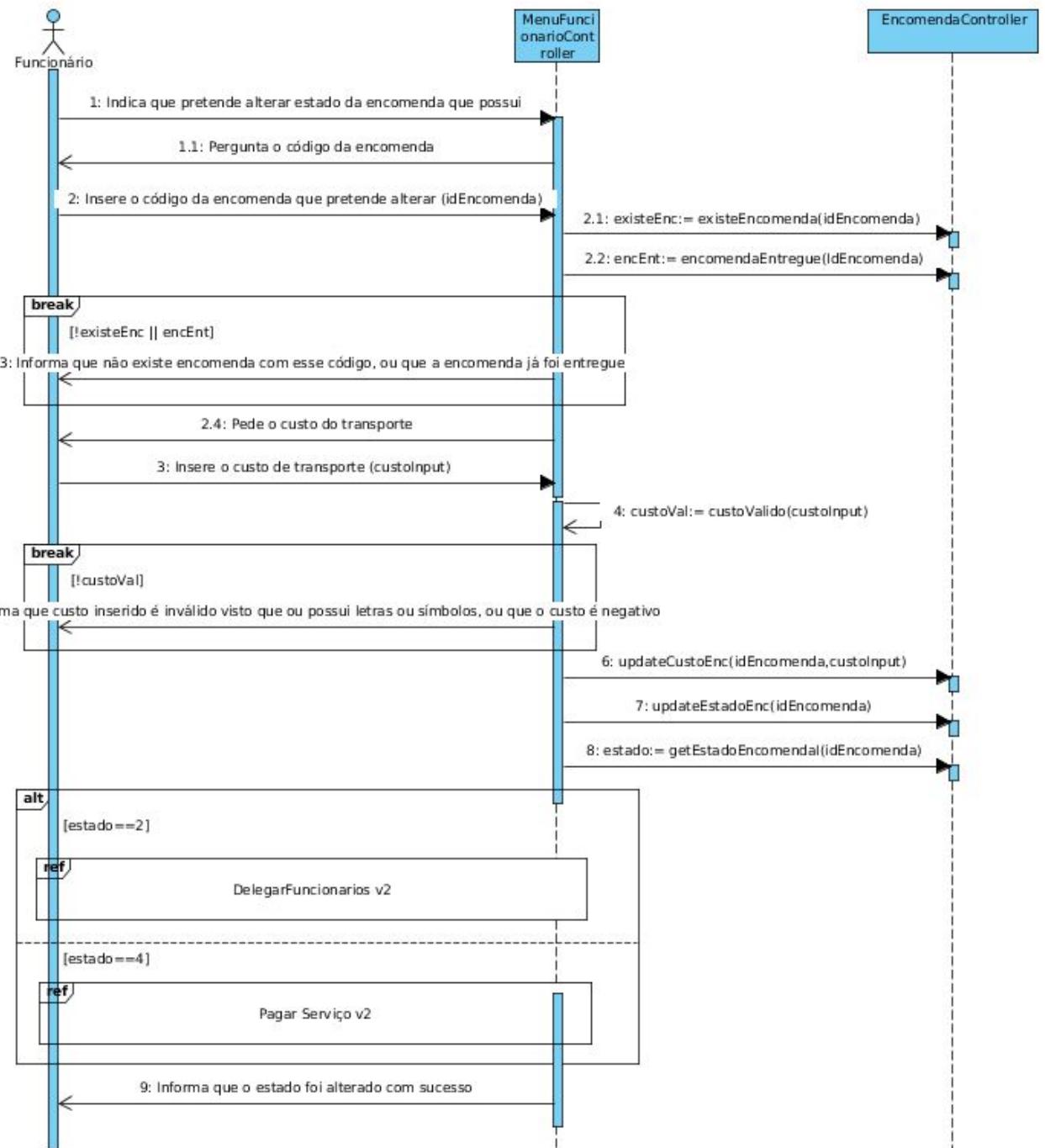
Diagramas de Sequência dos Use Cases (Definição de funções)

sd Alterar Dados v2

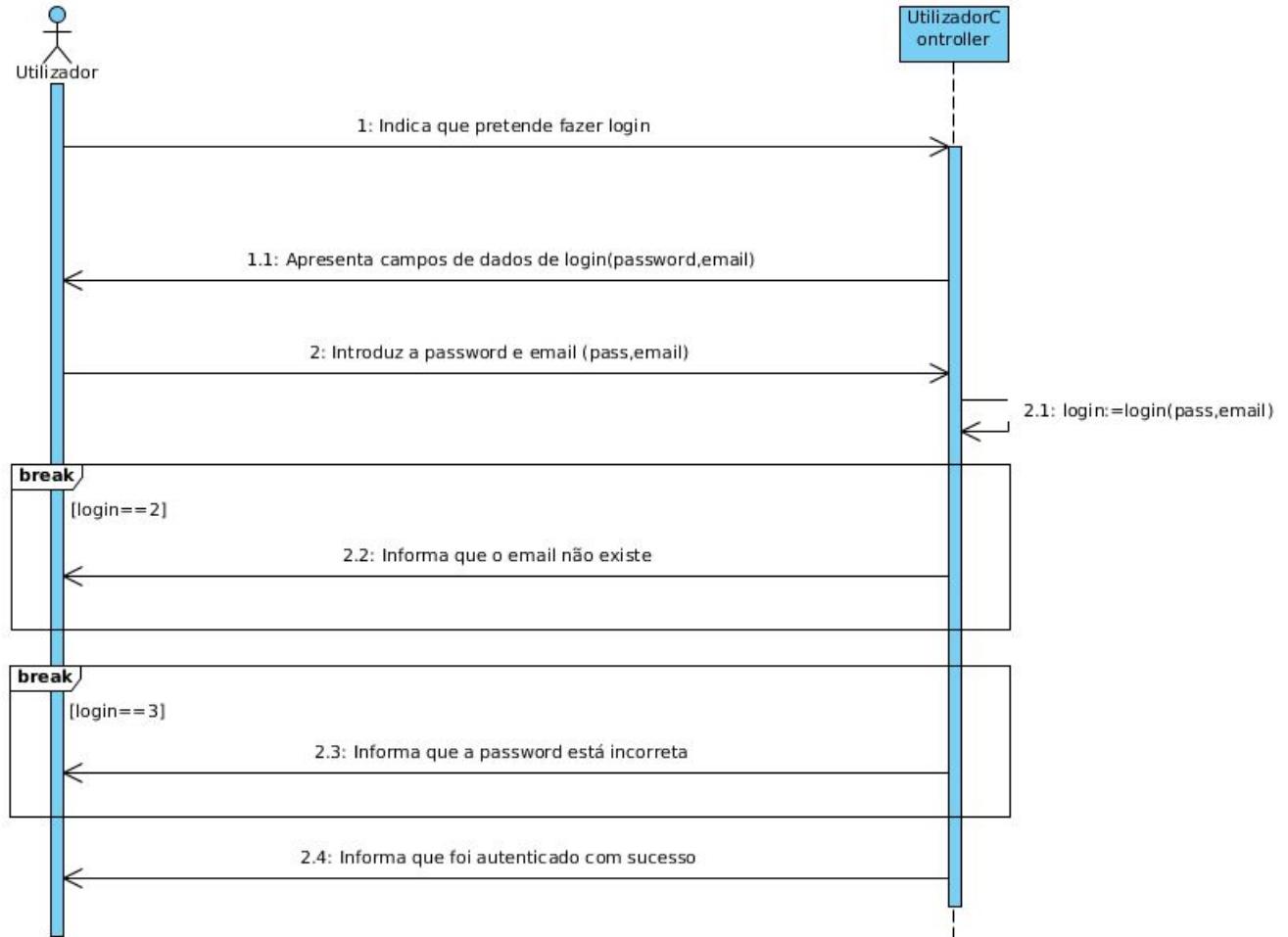




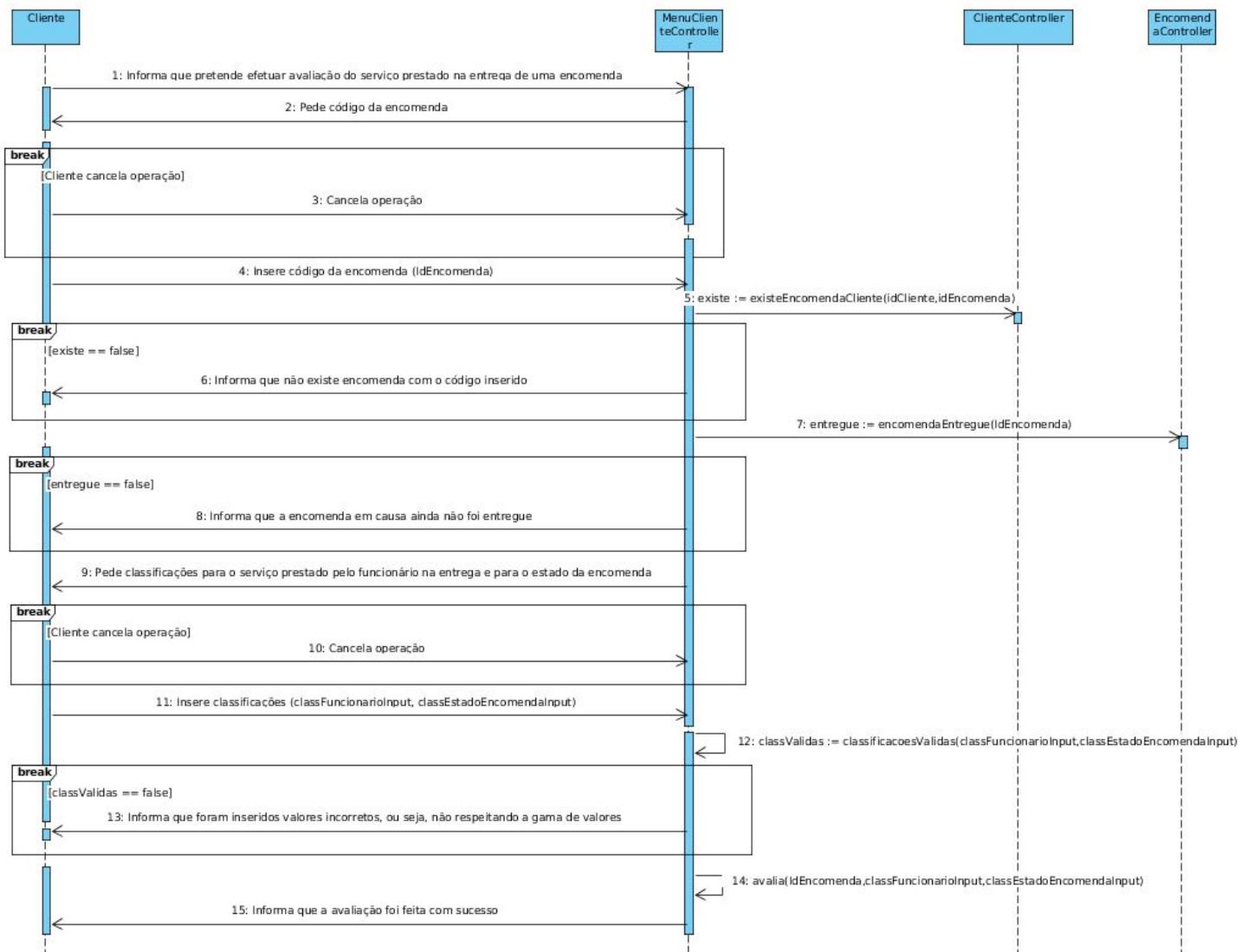
sd Atualizar Estado v2



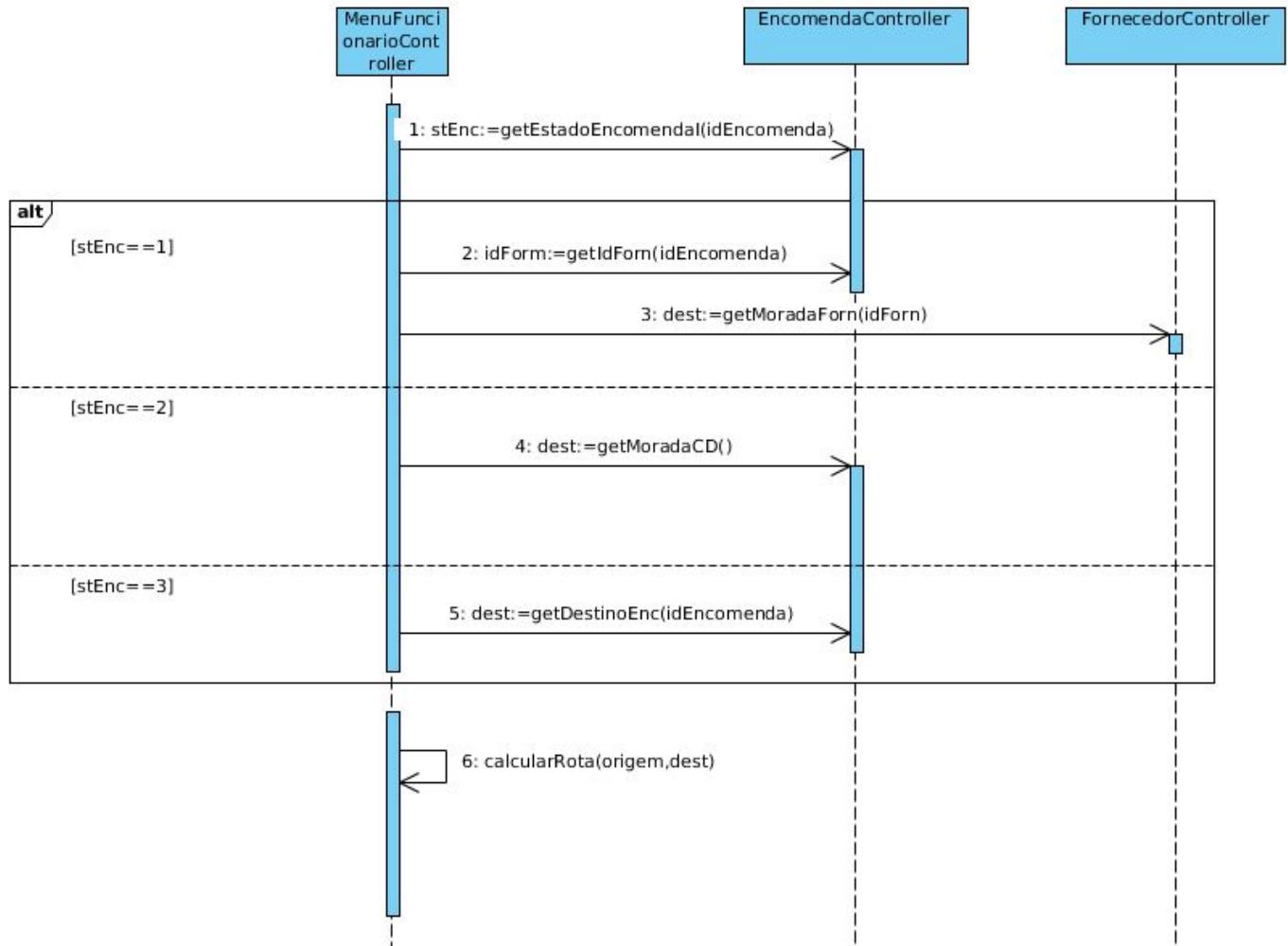
sd Autenticar v2



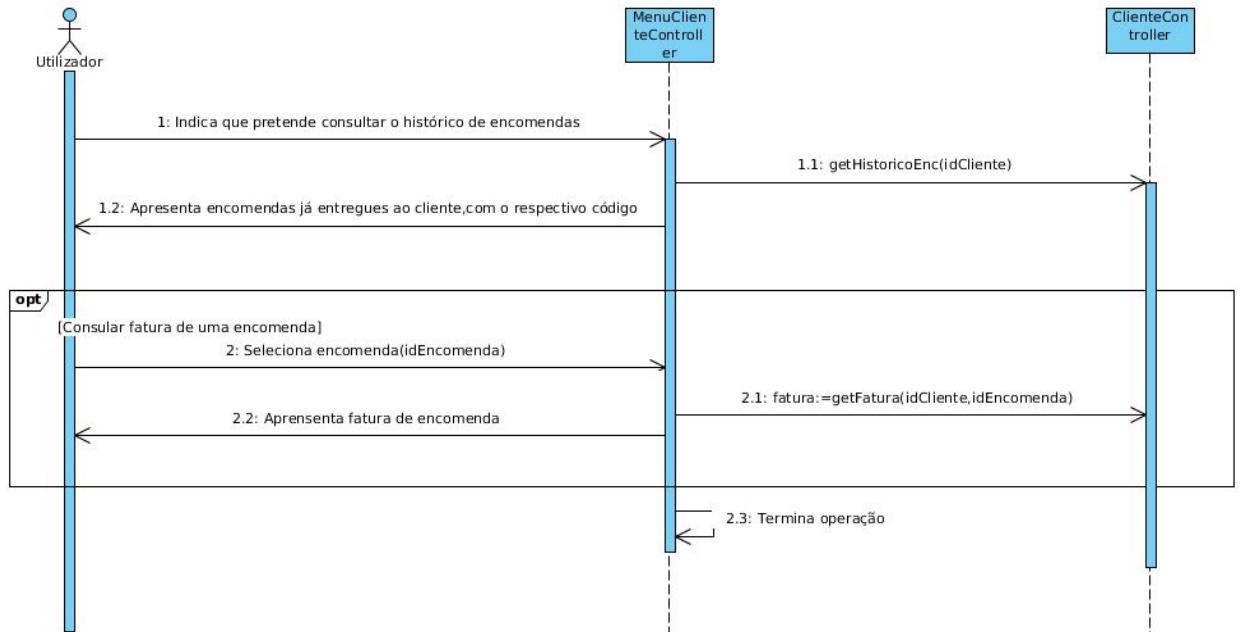
sd AvaliarServiço v2)



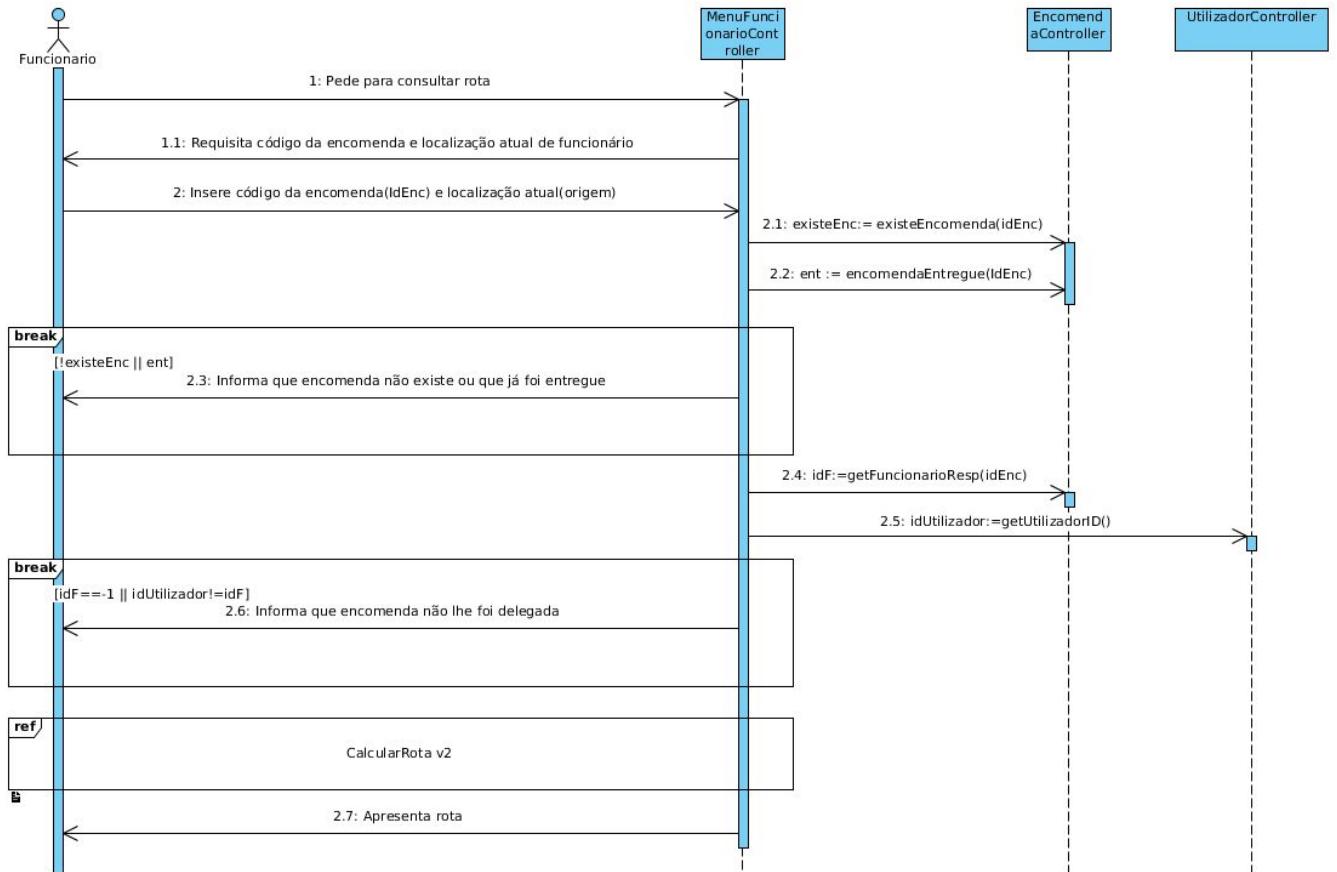
sd CalcularRota v2



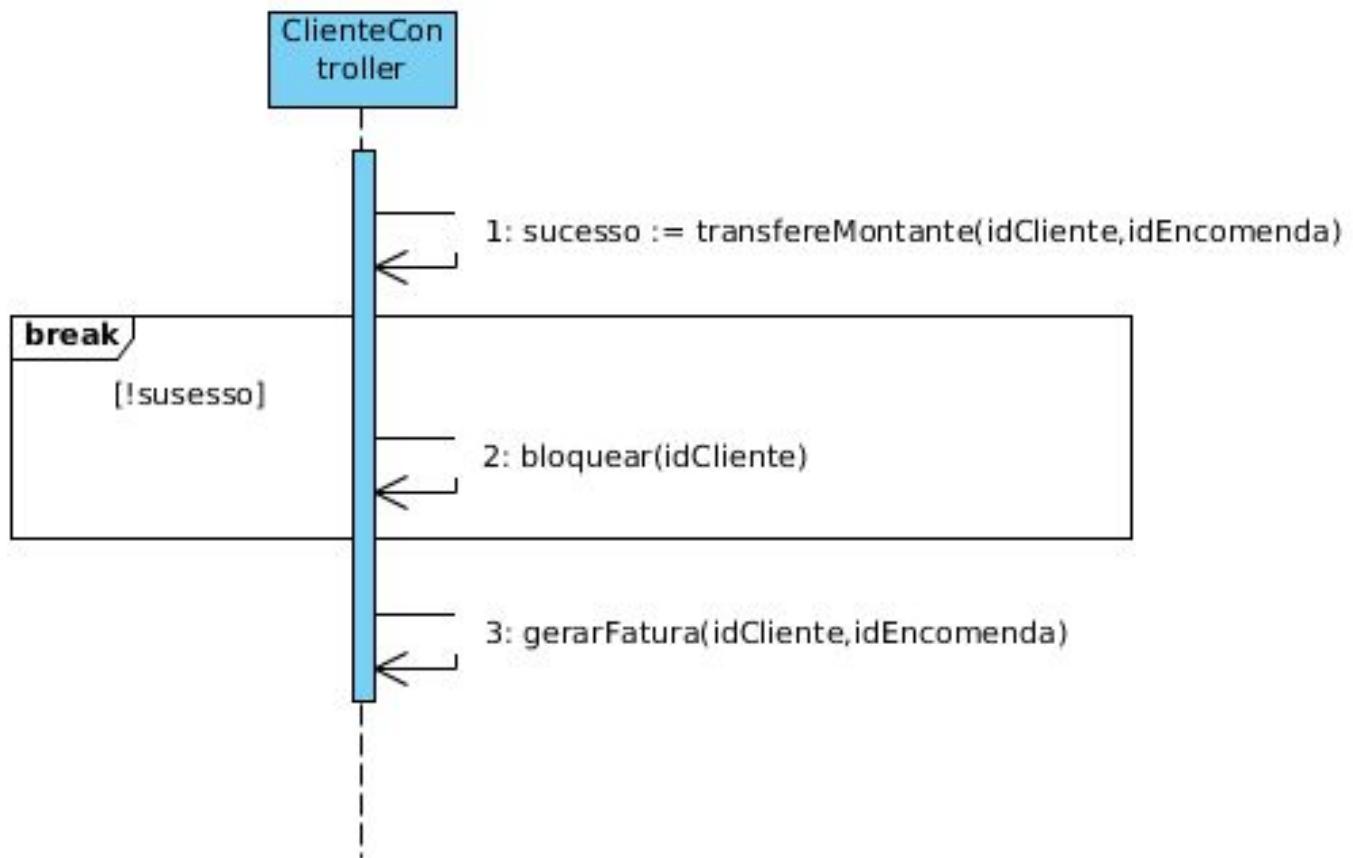
sd Consultar histórico v2



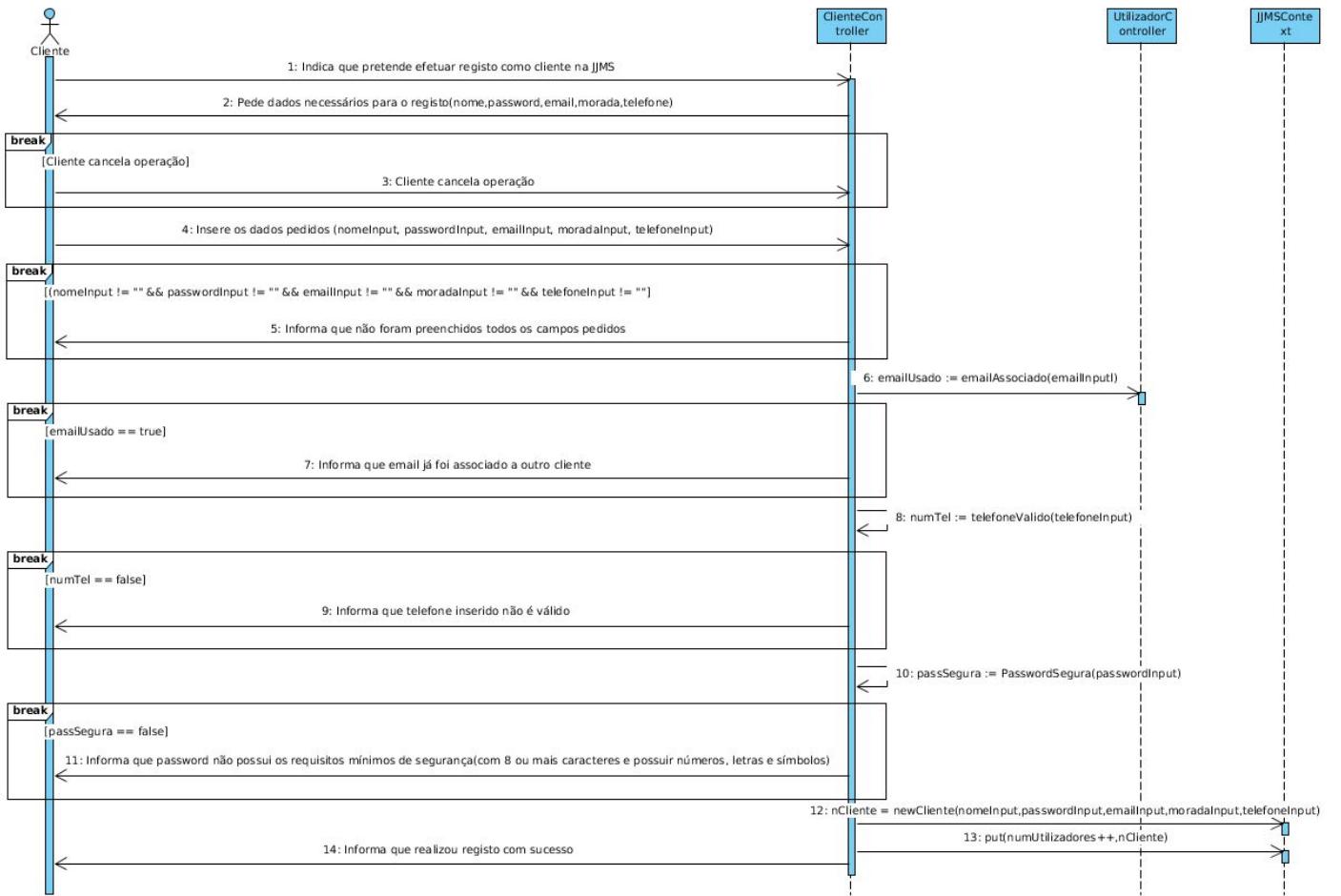
sd ConsultarRota v2



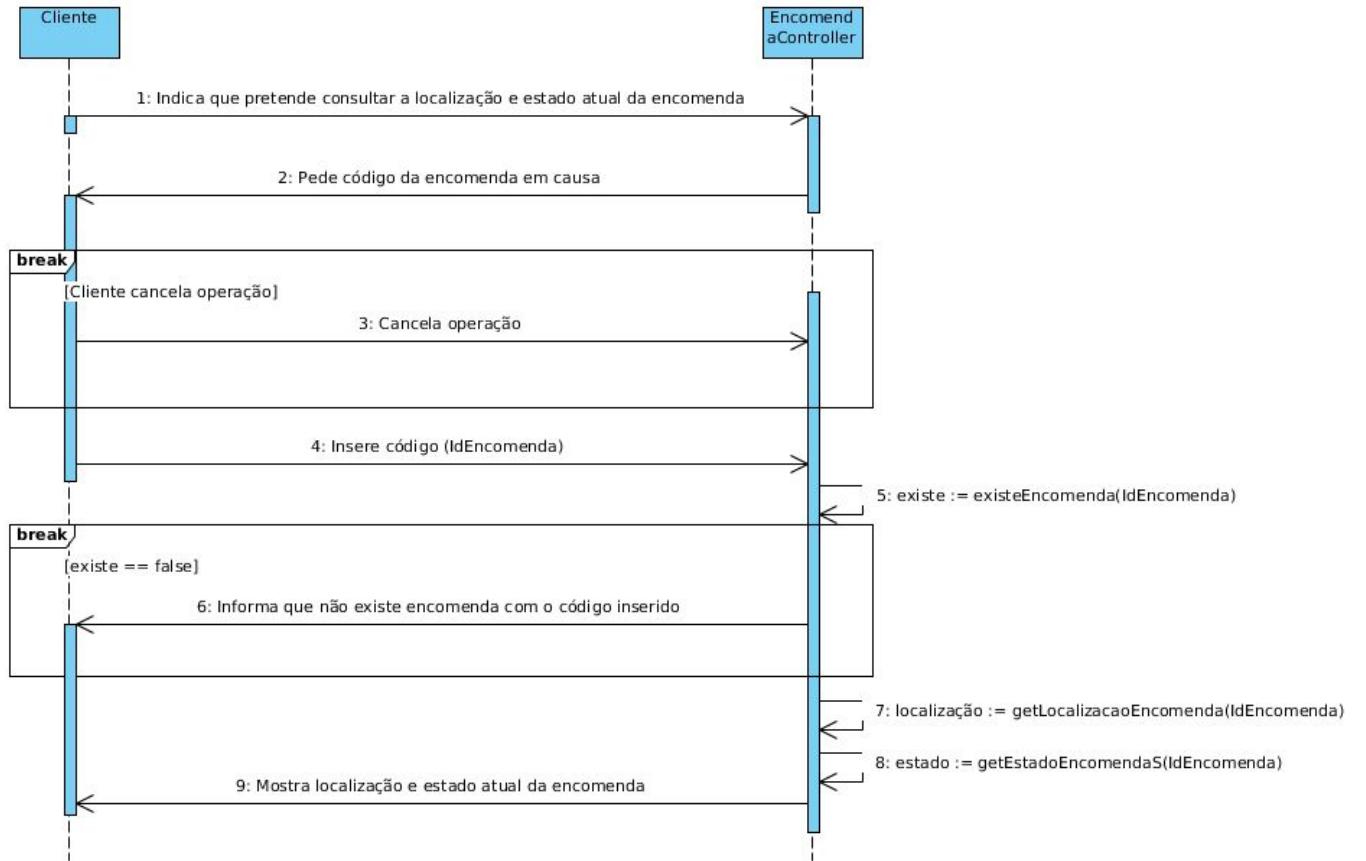
sd Pagar Serviço v2



sd Registrar v2

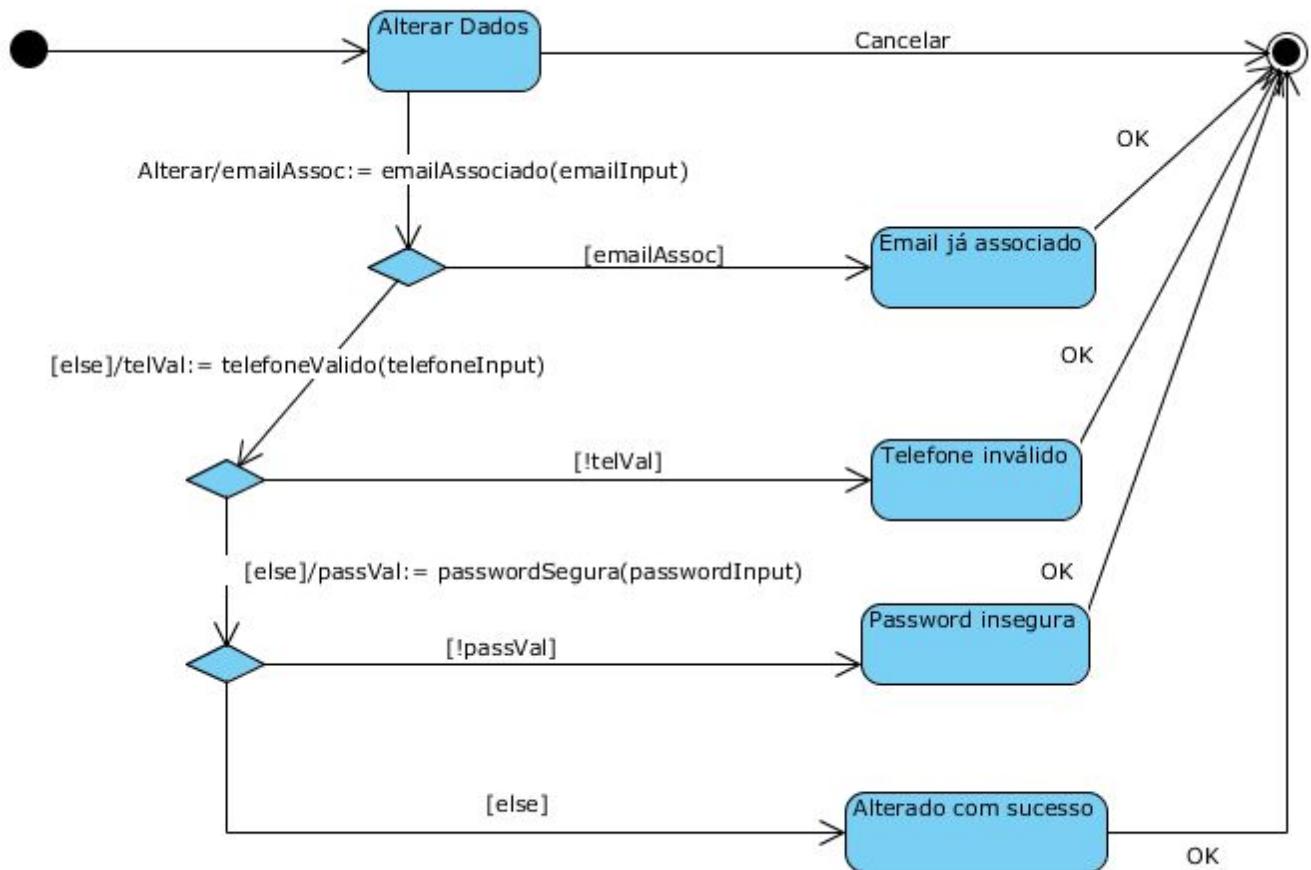


sd TrackingEncomenda v2

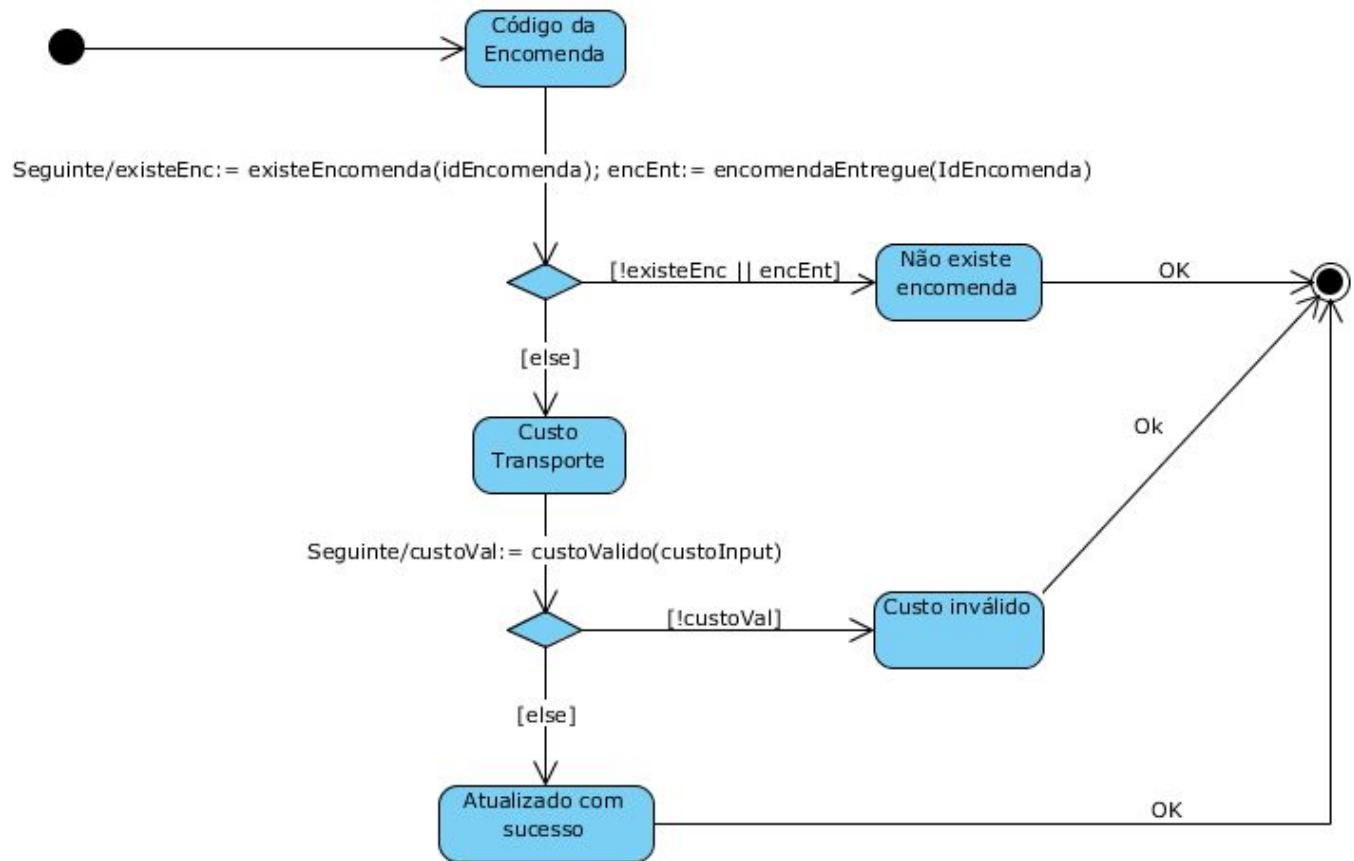


Máquinas de Estado da Prototipagem

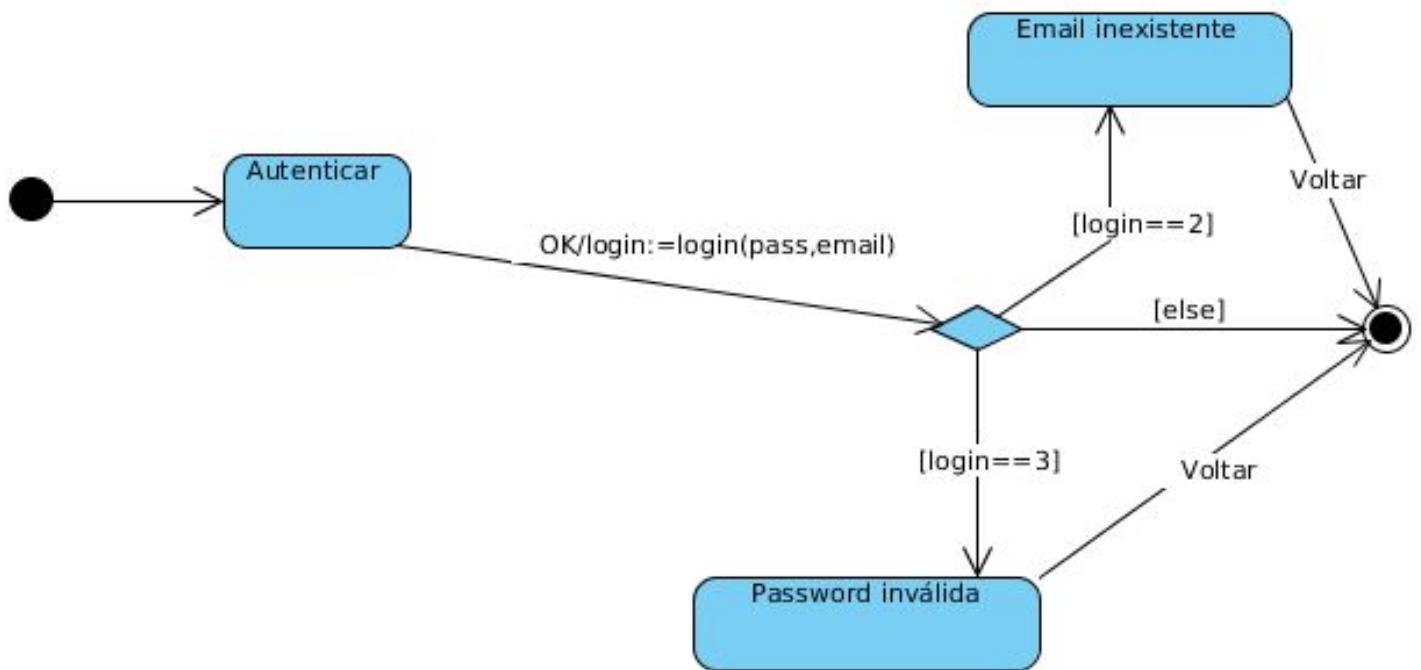
Alterar Dados



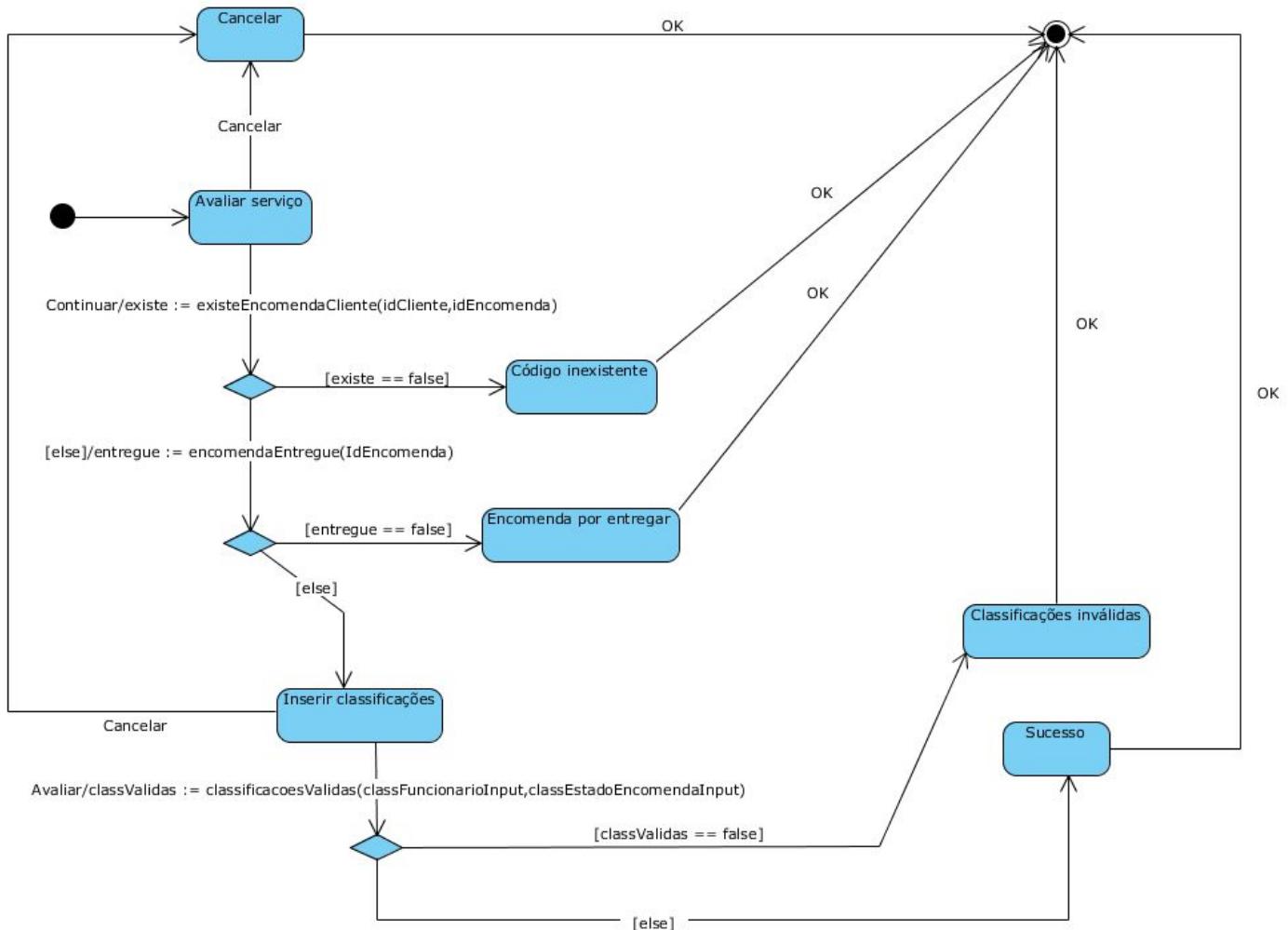
Atualizar Estado



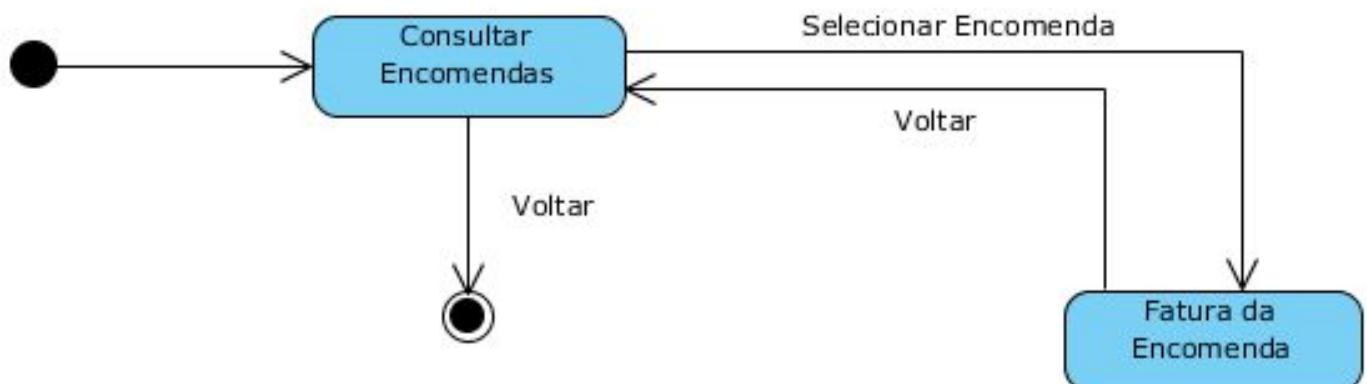
Autenticar



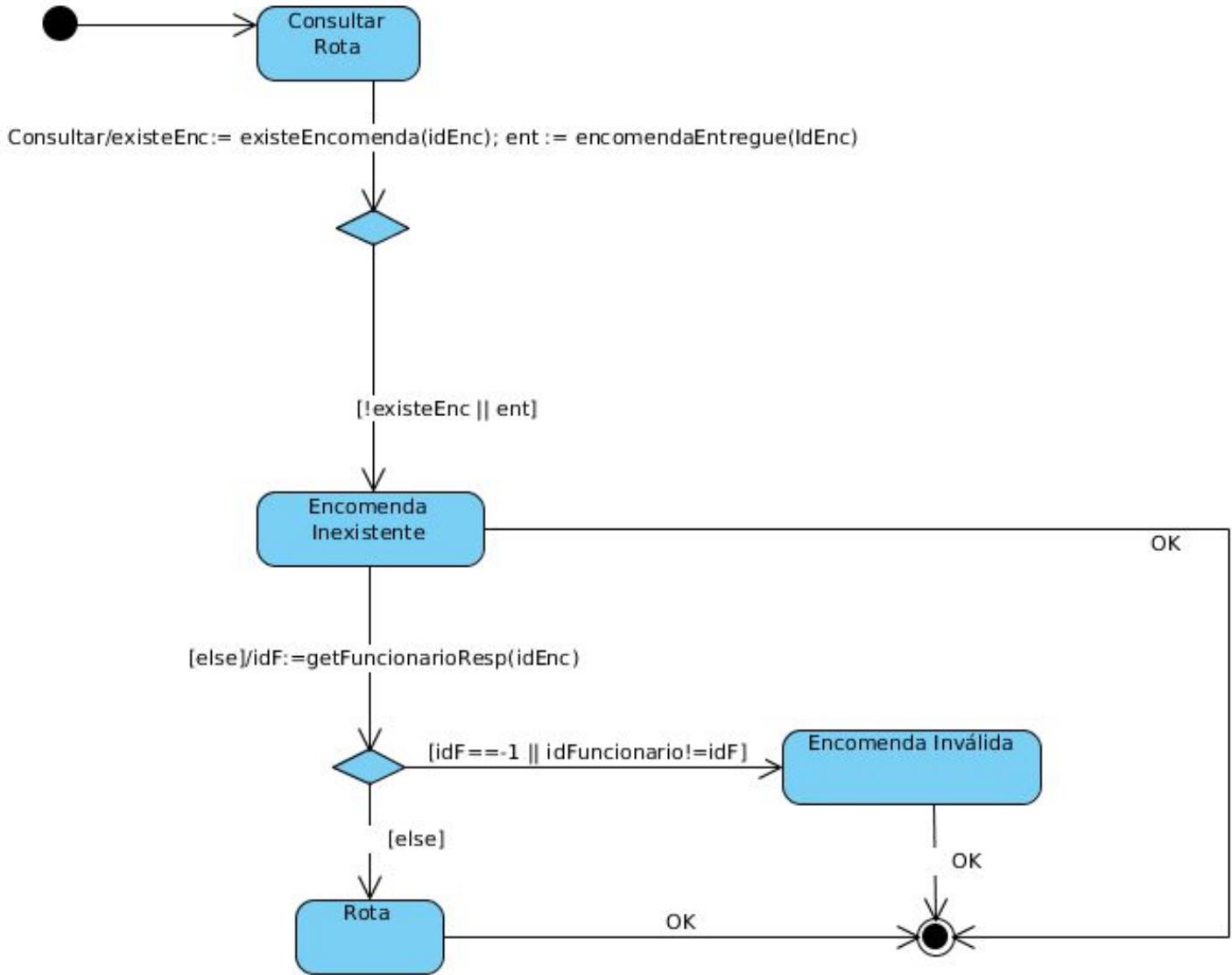
Avaliar Serviço



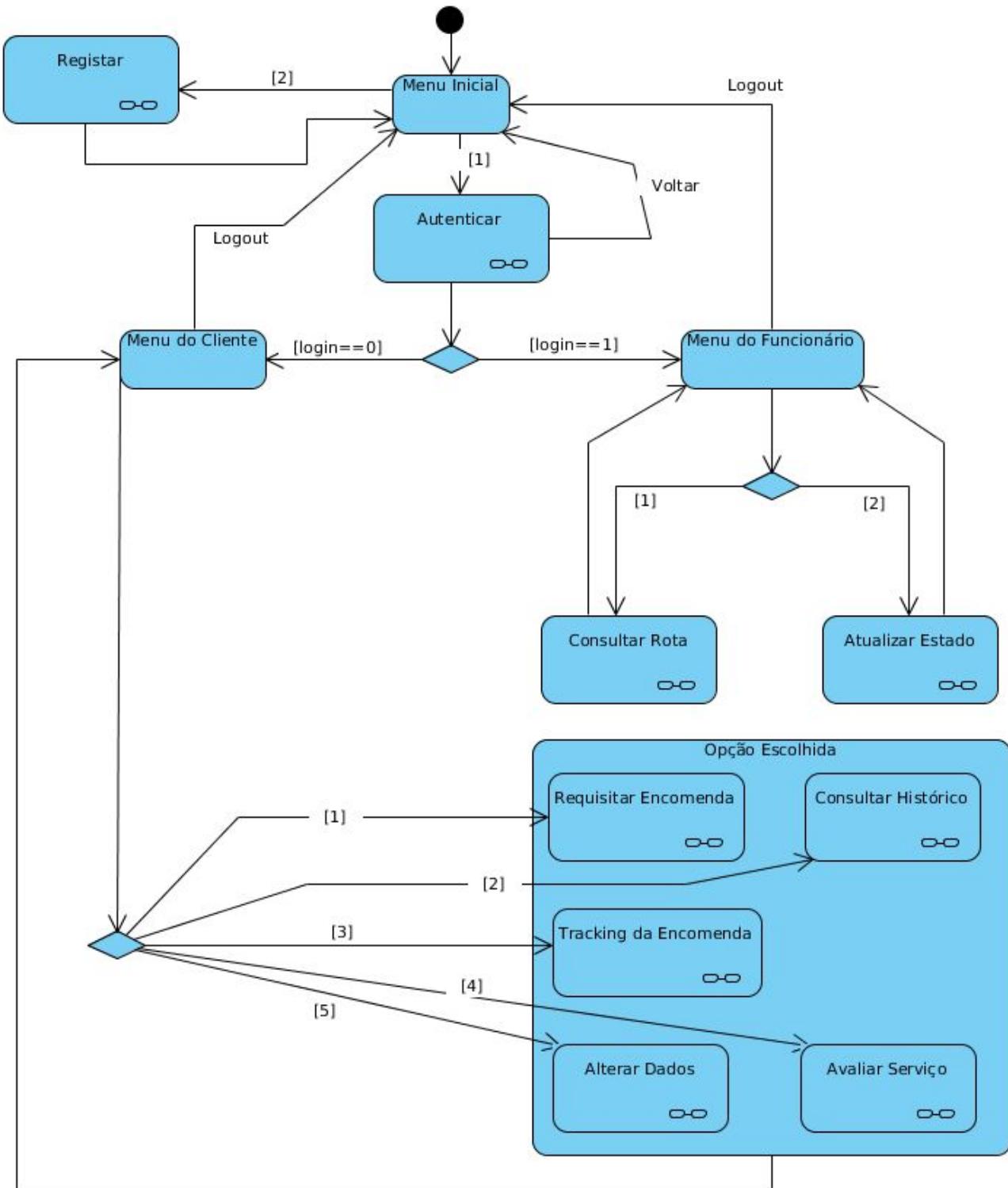
Consultar Histórico



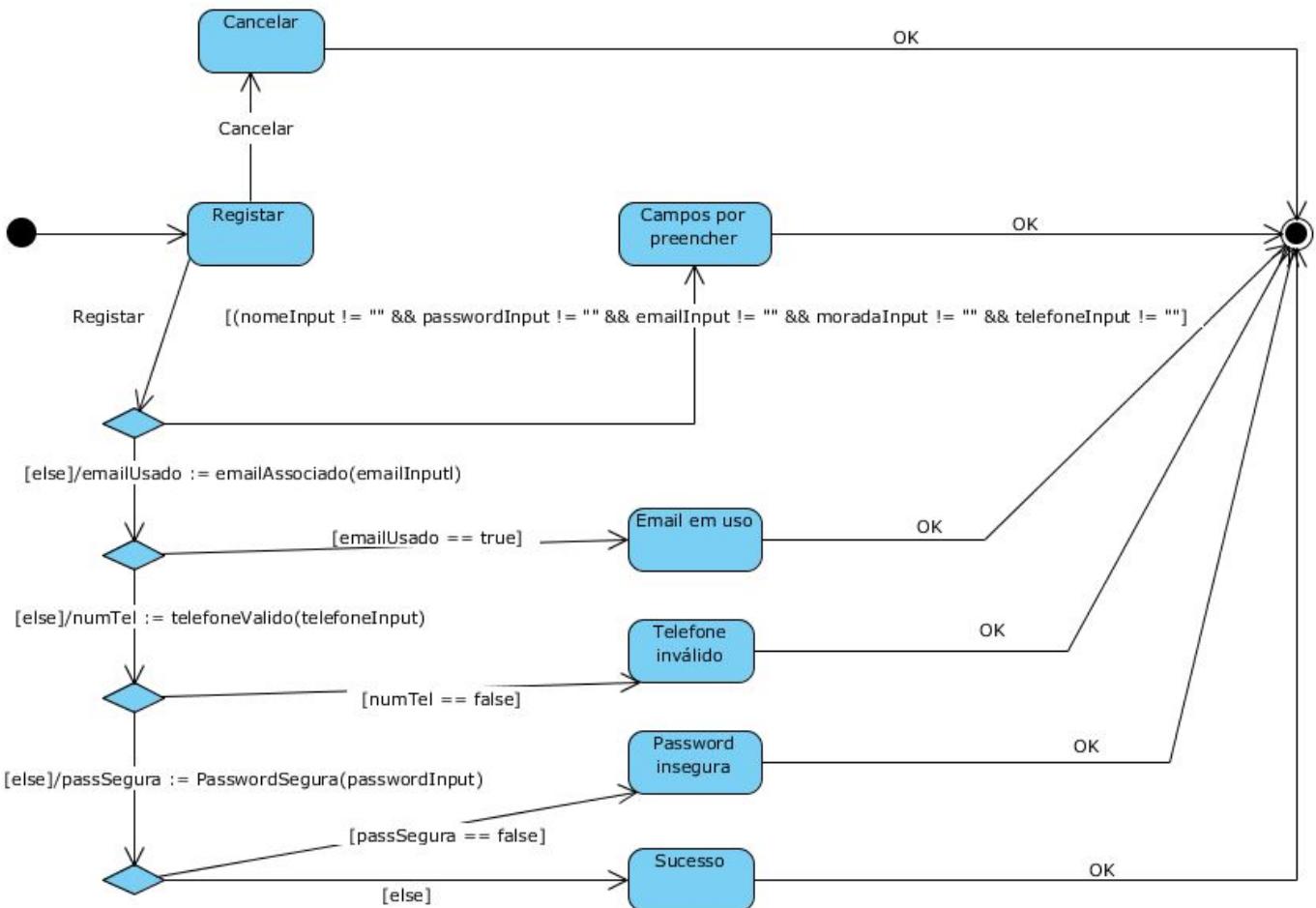
Consultar Rota



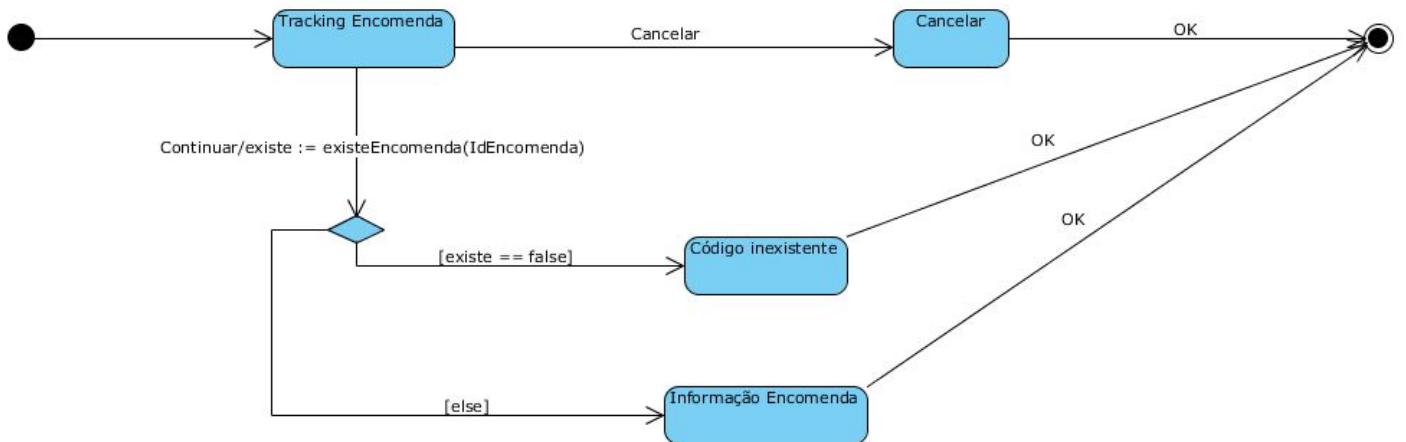
Geral



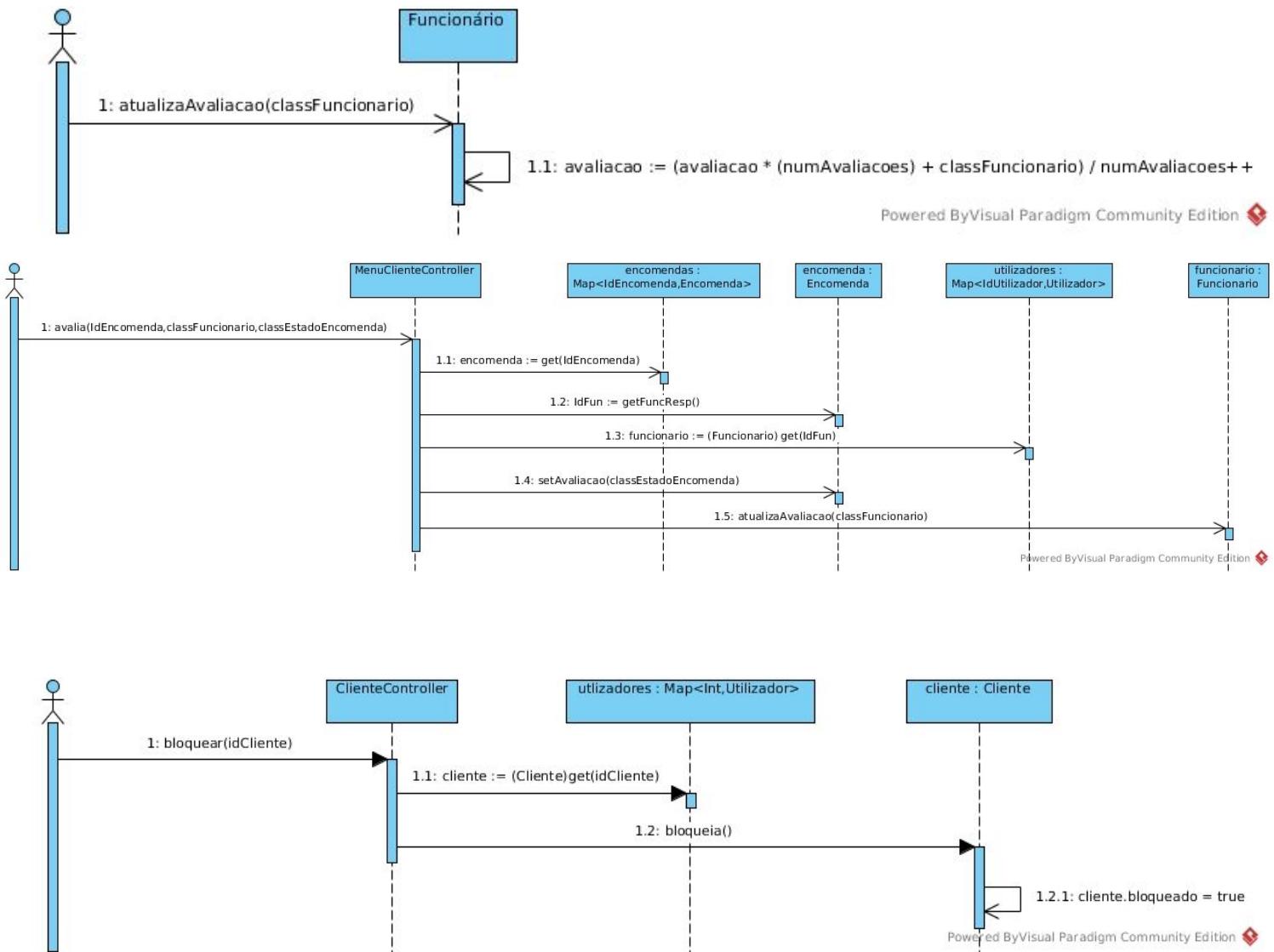
Registrar

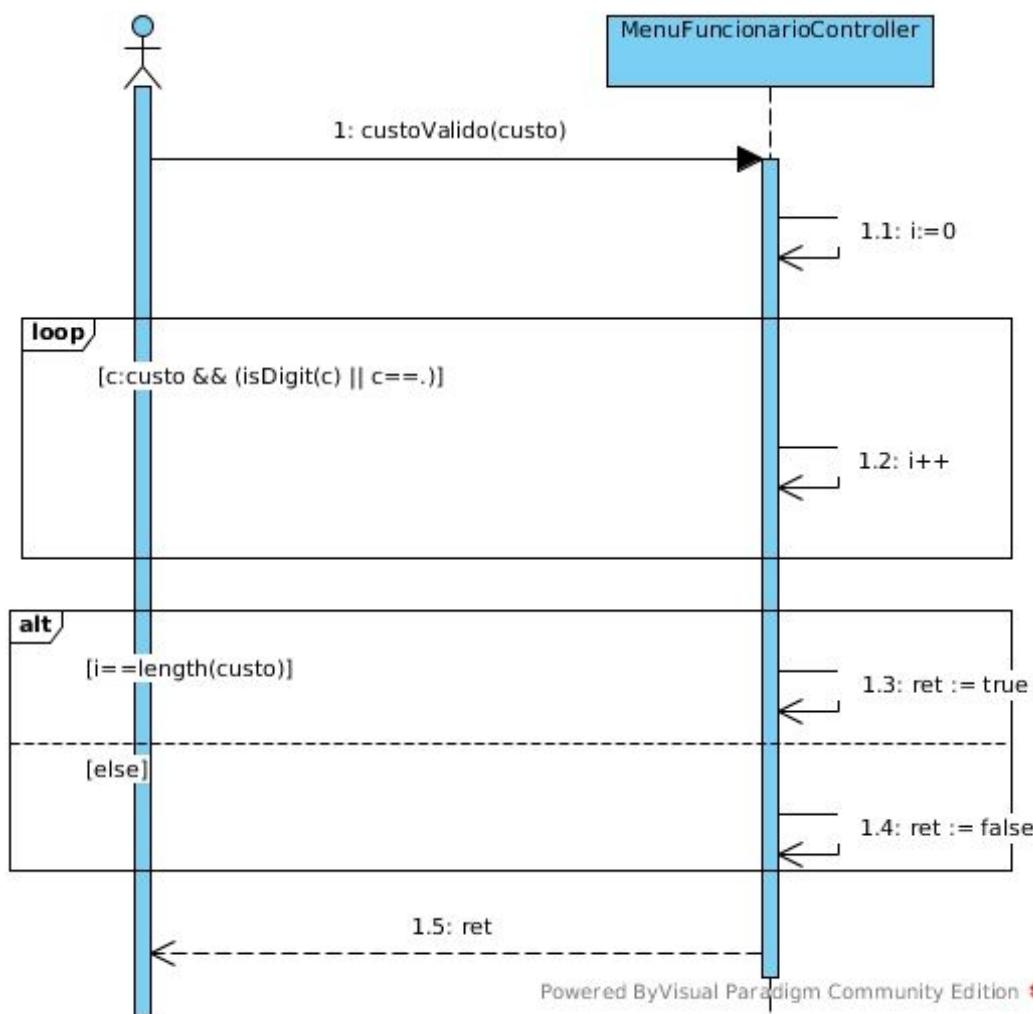
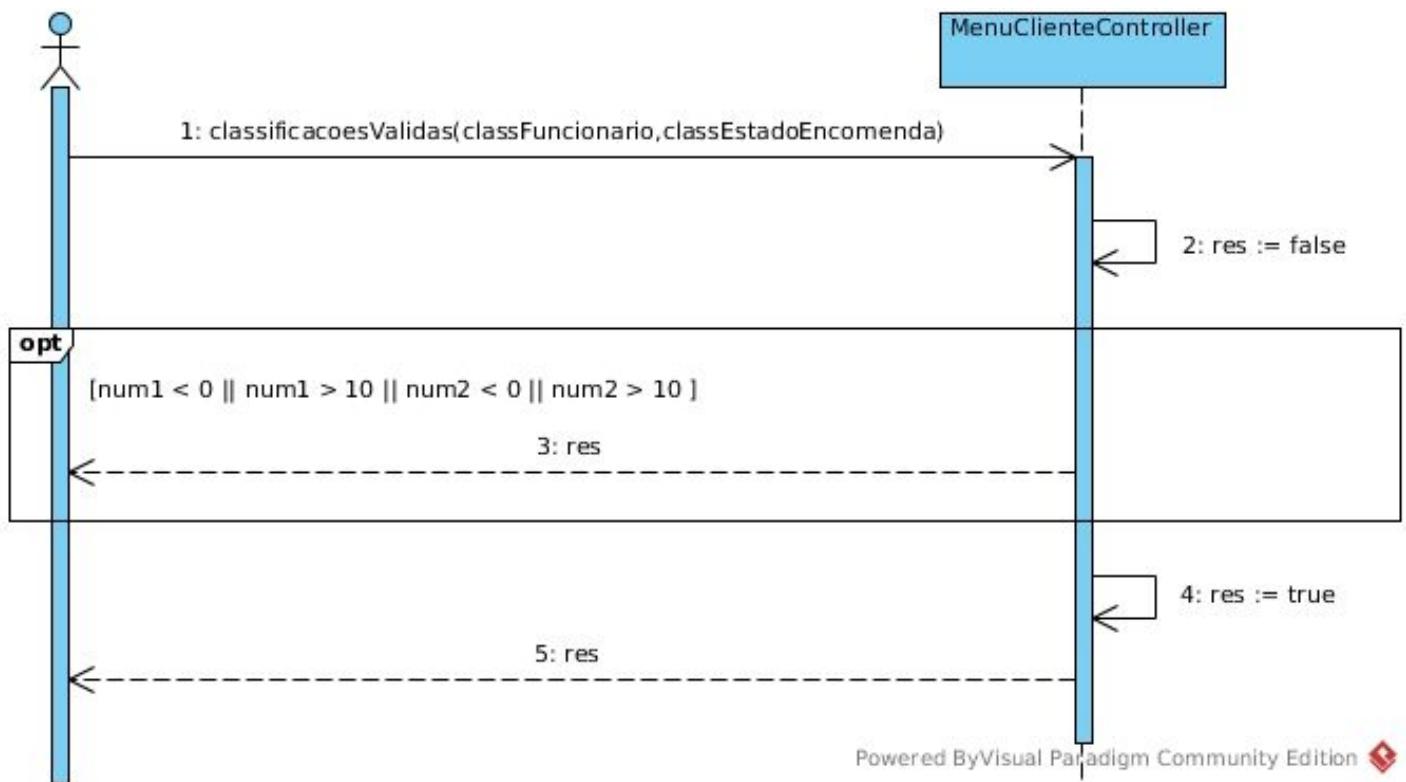


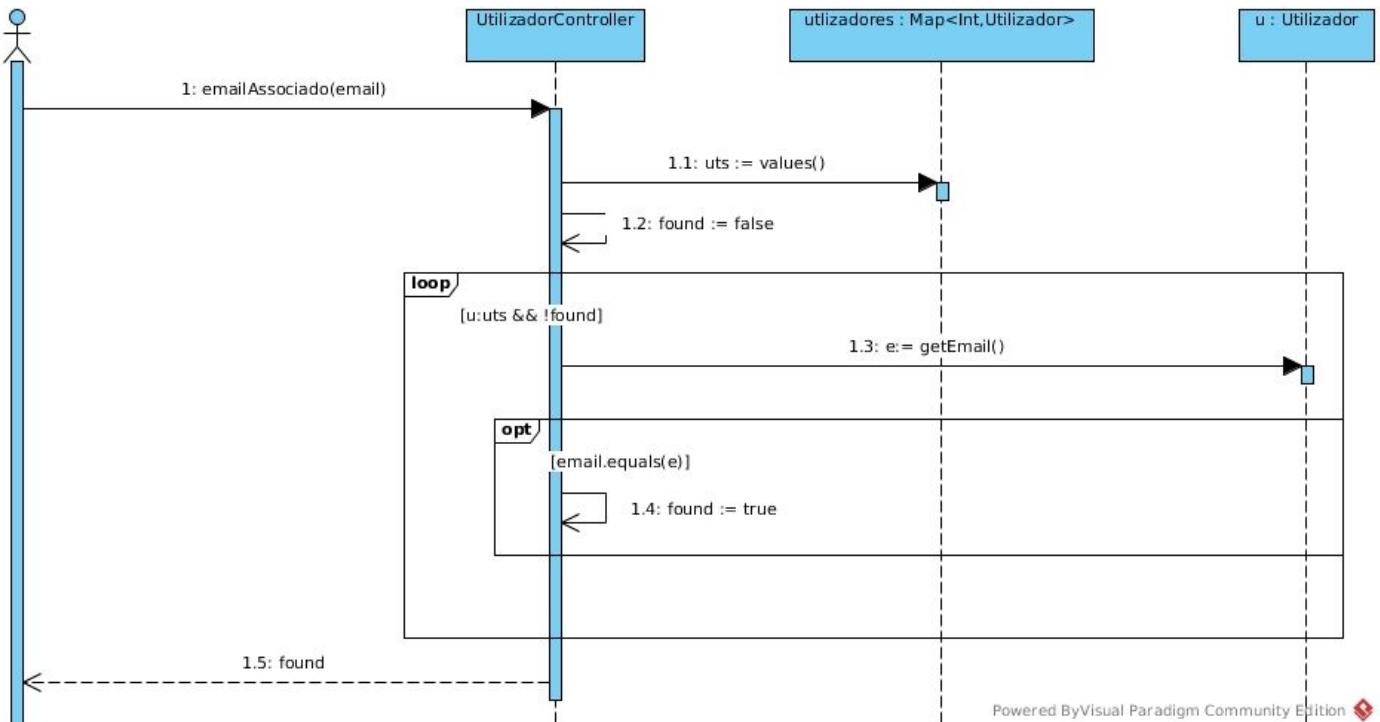
Tracking Encomenda



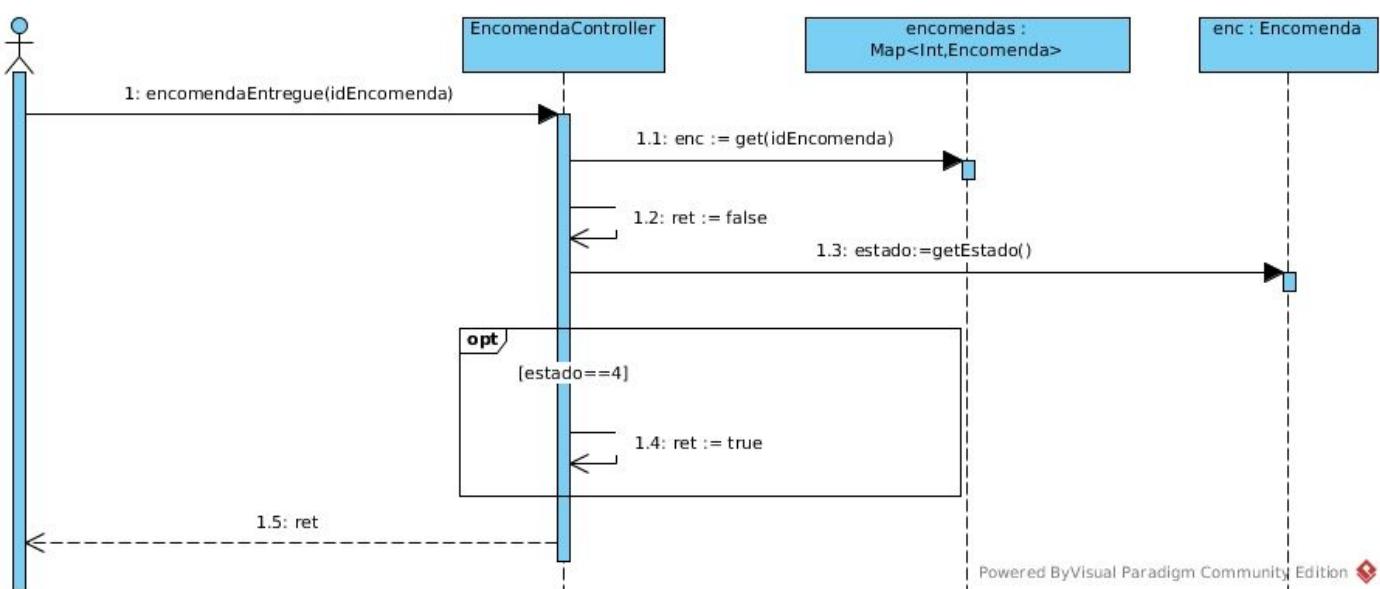
Diagramas de Sequência das funções definidas



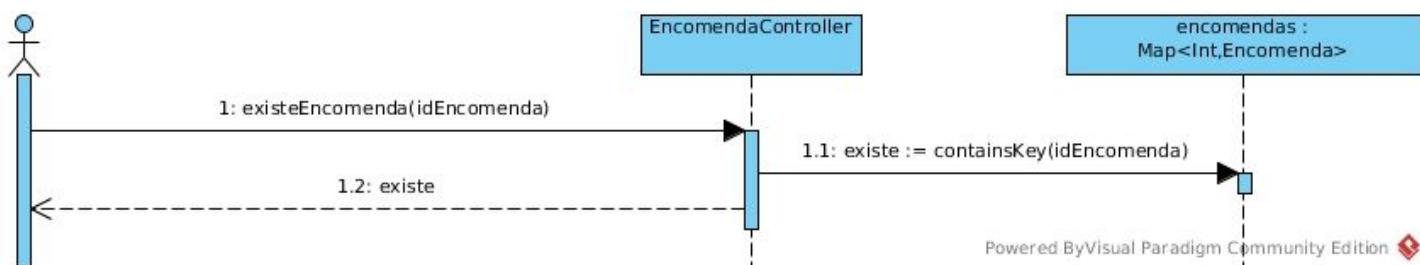




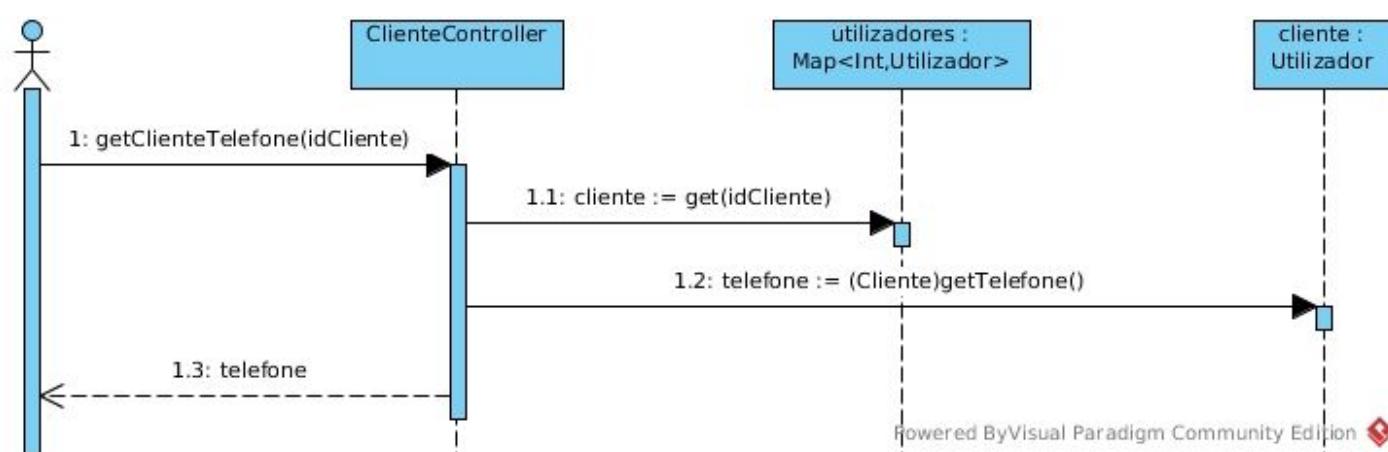
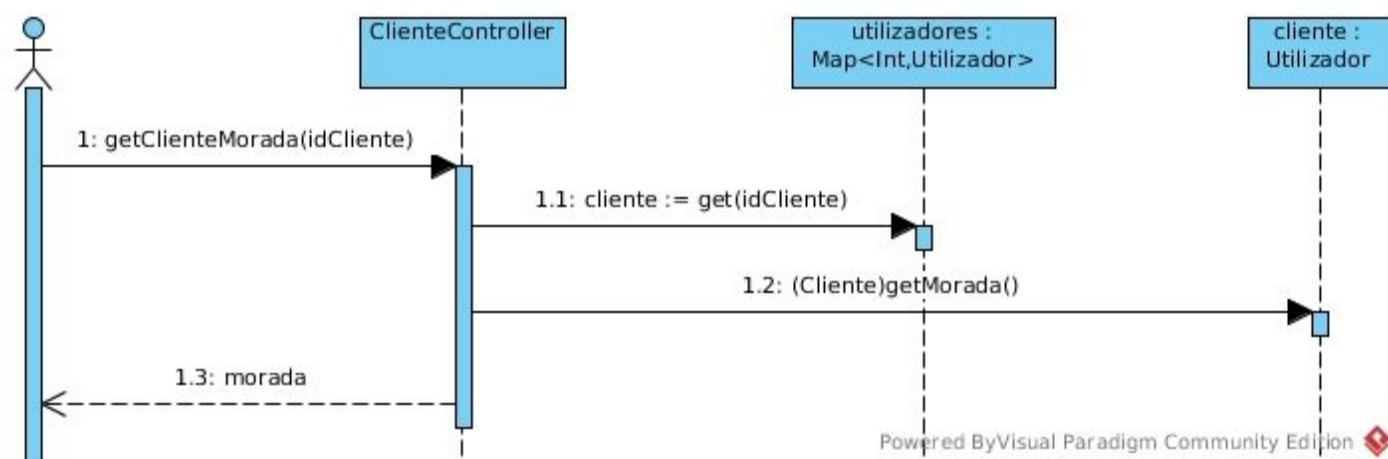
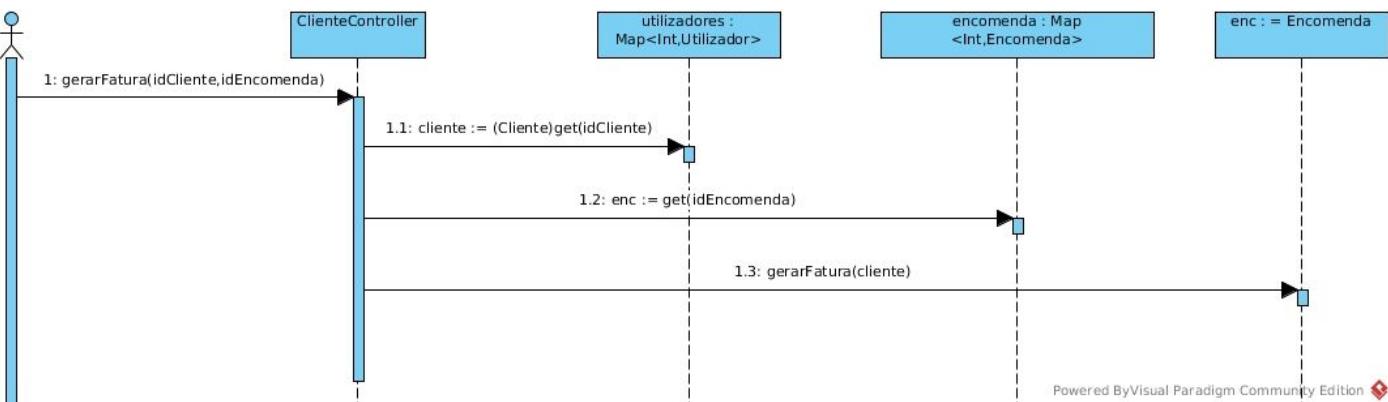
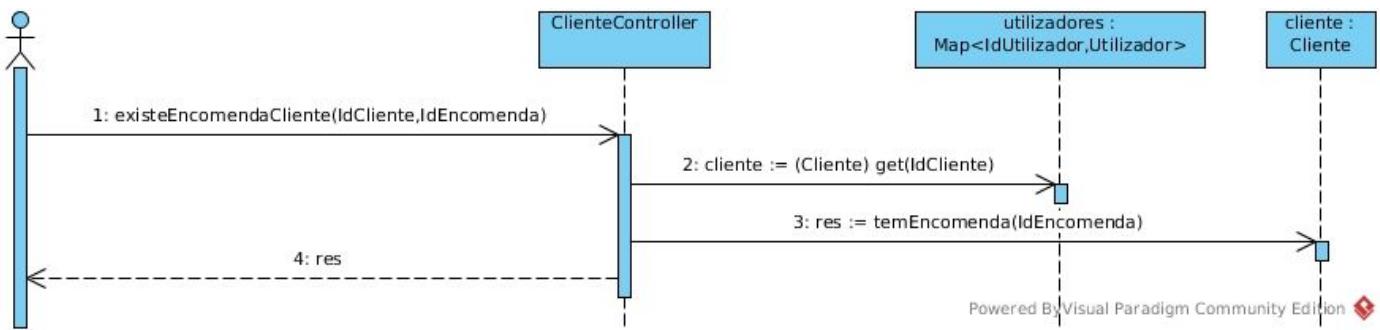
Powered ByVisual Paradigm Community Edition

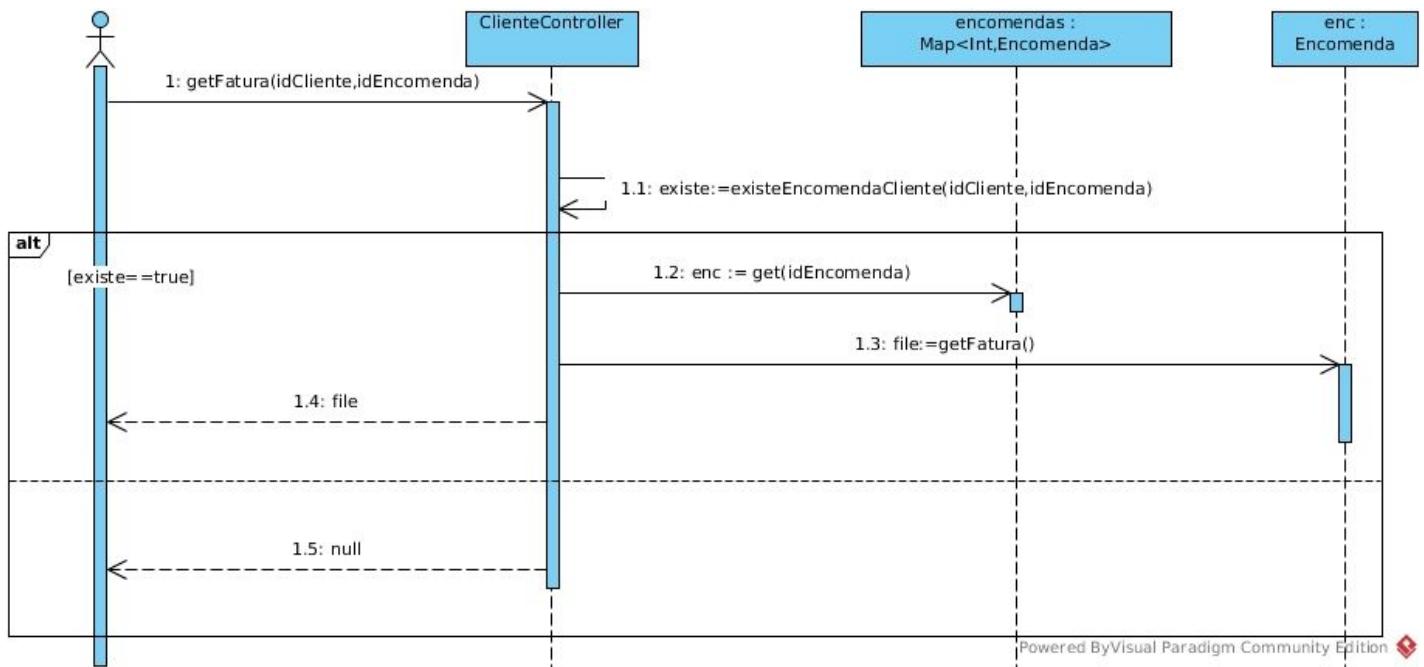


Powered ByVisual Paradigm Community Edition

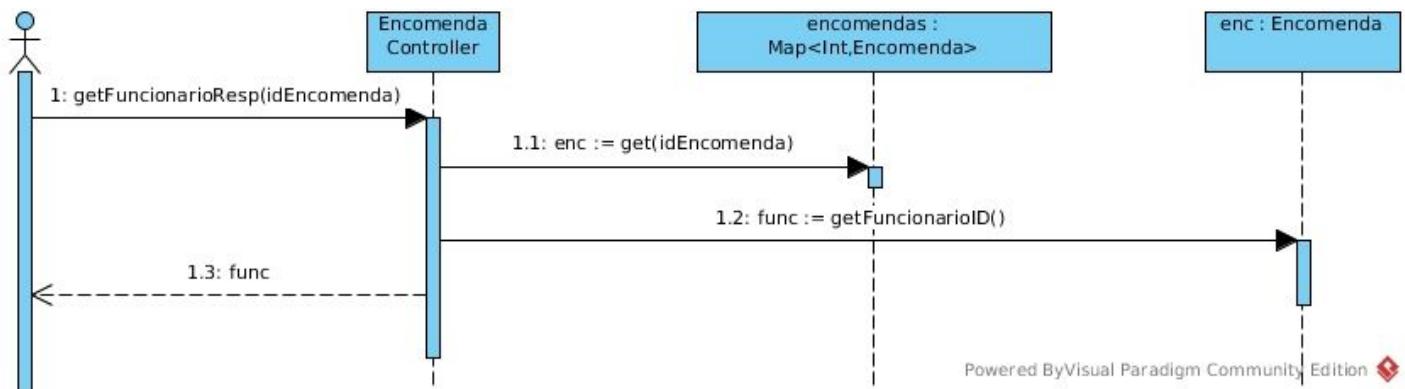


Powered ByVisual Paradigm Community Edition

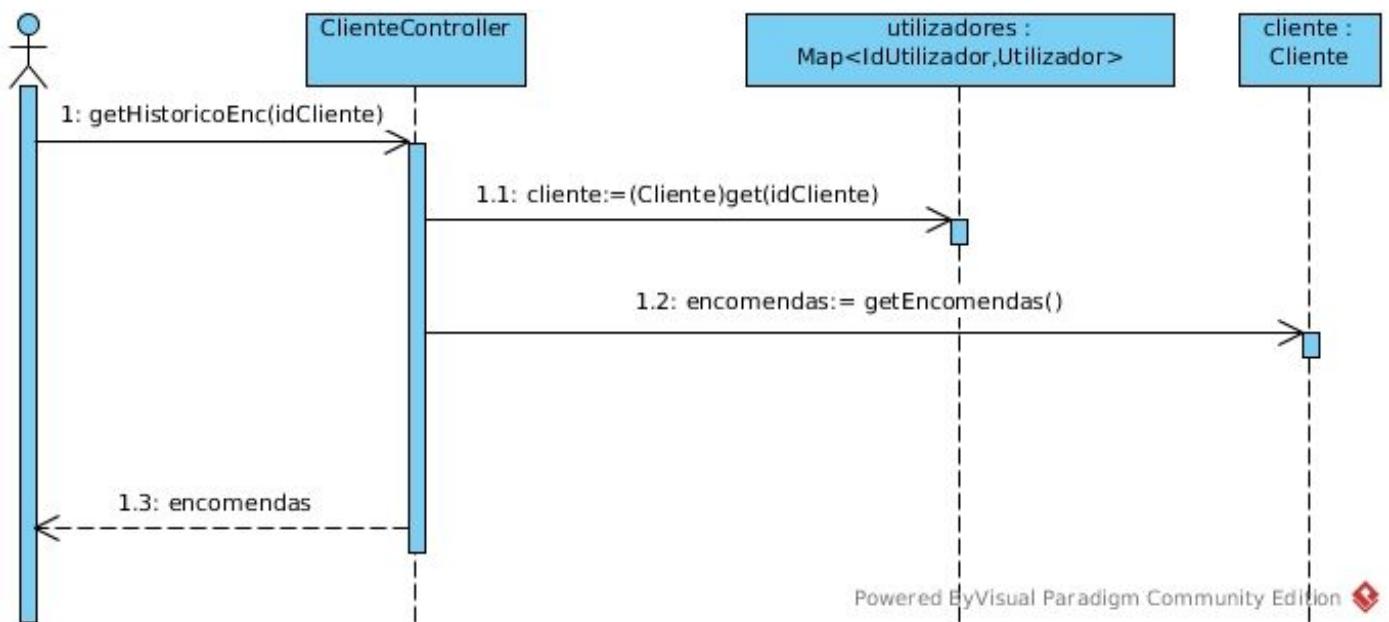




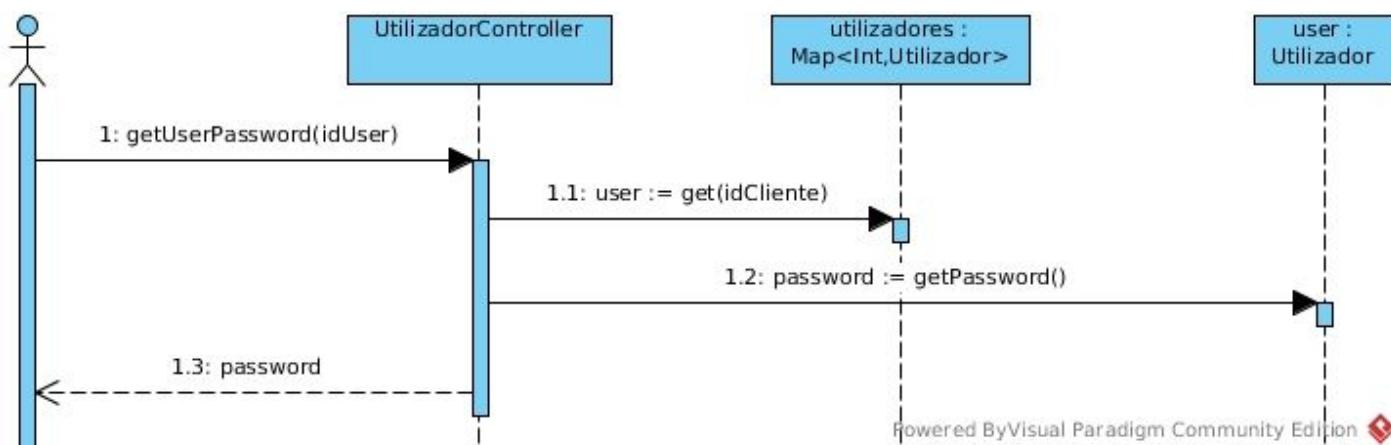
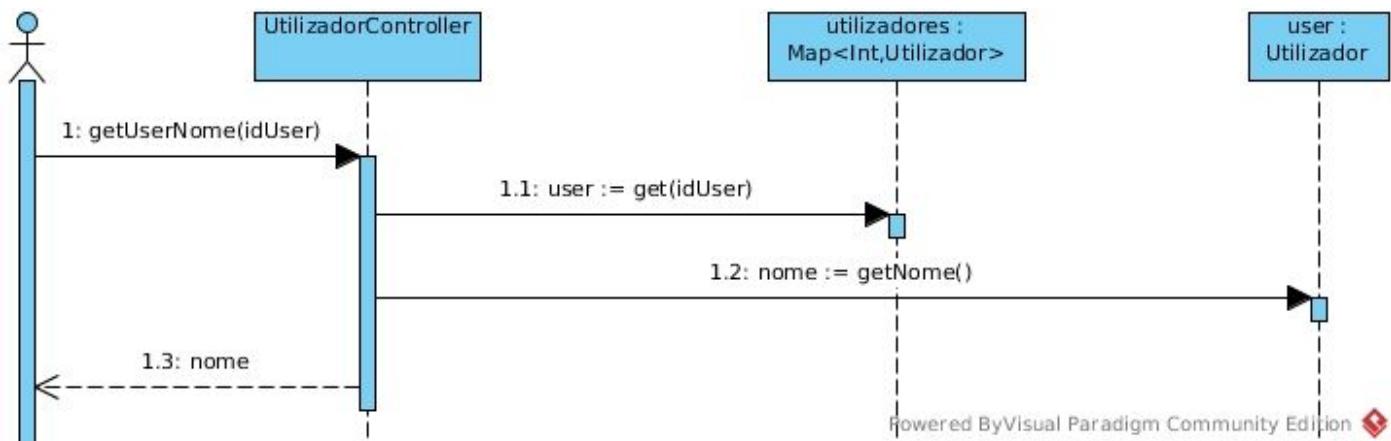
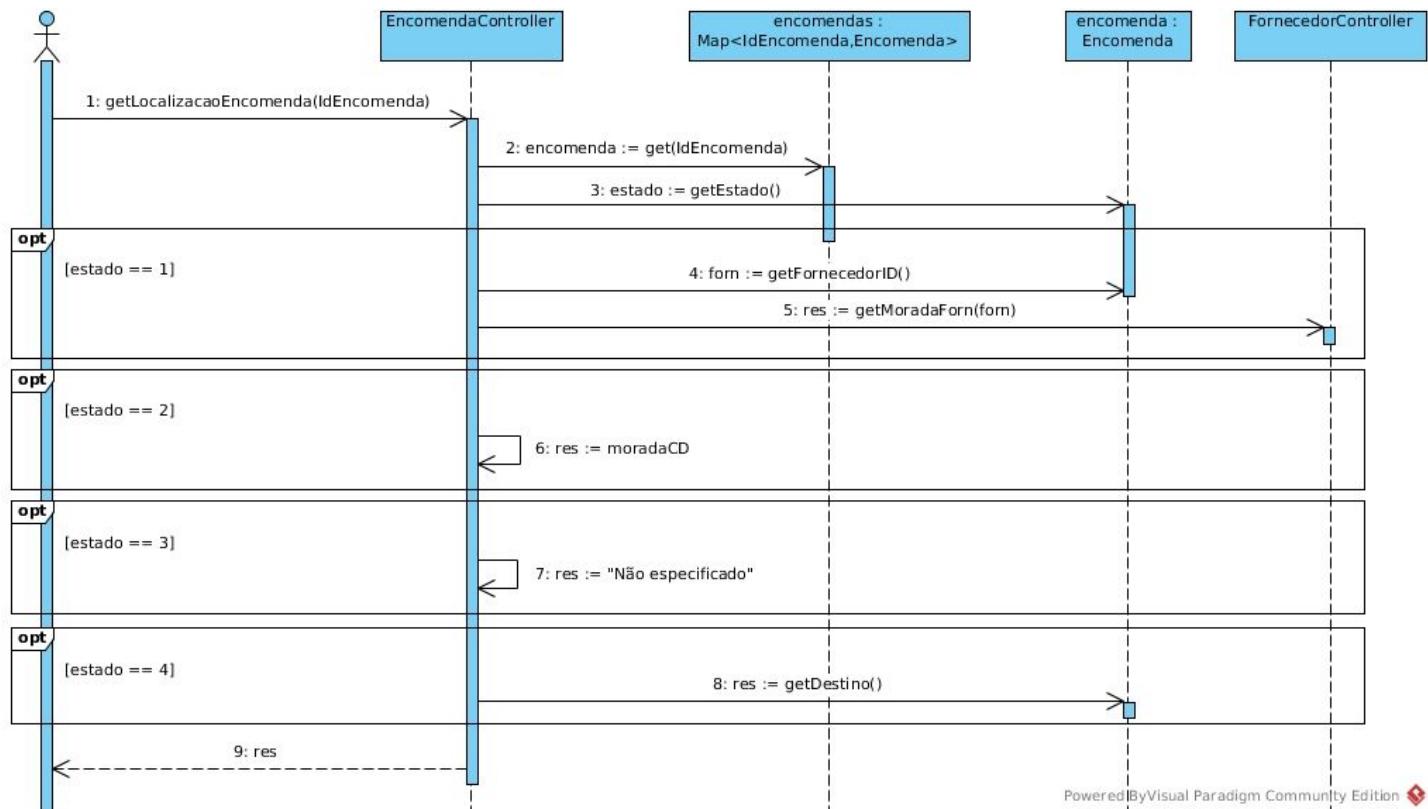
Powered ByVisual Paradigm Community Edition

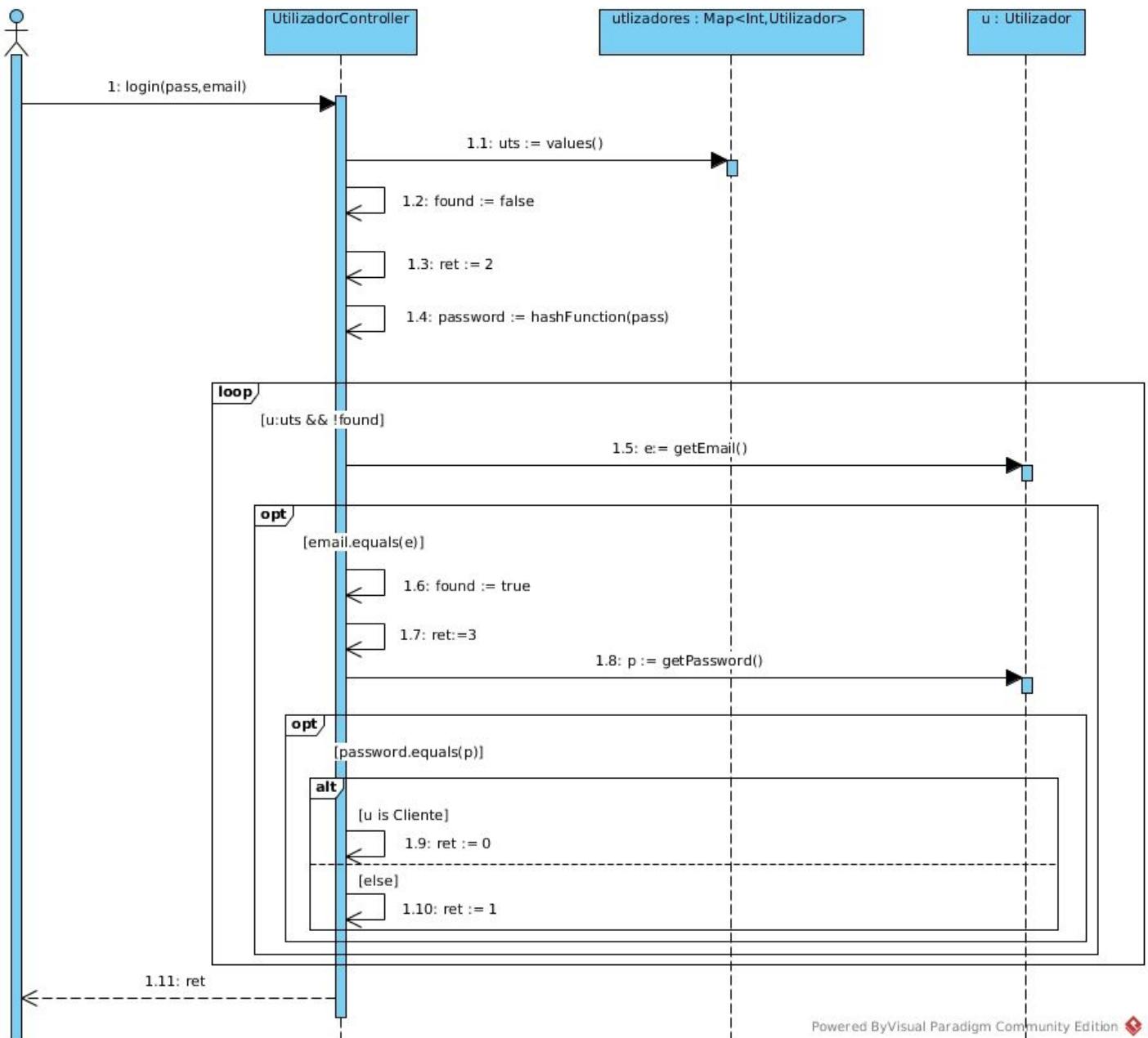


Powered ByVisual Paradigm Community Edition



Powered ByVisual Paradigm Community Edition





Powered By Visual Paradigm Community Edition

