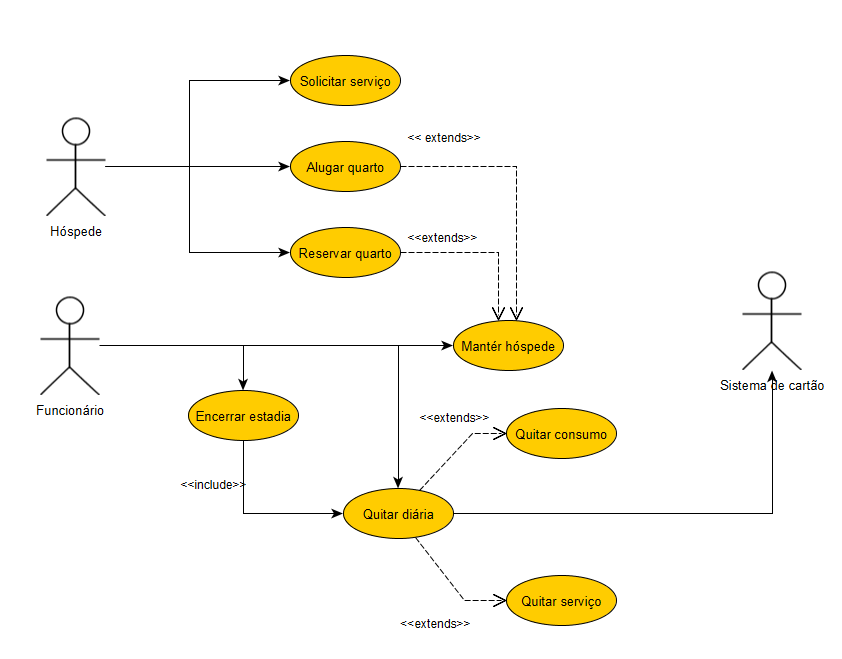
**Engenharia de Software 3 – Lista 1**

Integrantes:

|  |  |
| --- | --- |
| Nome | RA |
| Guilherme Ishii | 151.058-4 |
| Lucas Gigek Carvalho | 142.065-8 |
| Renan Luiz Cardoso | 132.074-2 |
| Renato Geraldes Eraclide | 122.073-1 |
| Ricardo de Lucas do Nascimento | 151.071-1 |

**1 – Construa um diagrama de casos de uso.**



**2 – Especifique o CSU1 apresentando os fluxos (cenários) principal, alternativo e de exceção. Utilizar um template que seja inteligível. Os casos de uso <<extend>> e <<include>> (se houver) podem ser especificados junto com o caso de uso principal.**

CSU01 – Reservar Quarto

**Sumário:** Hóspede usa o sistema para reservar um quarto

**Ator primário:** Hóspede

**Requisitos Funcionais:** RF01

**Fluxo principal:**

1. Sistema exibe a tela de Reserva de Quartos;
2. Sistema solicita a entrada de dados;
3. O ator informa os seguintes dados para o sistema: [E1]
   1. Data de início da hospedagem;
   2. Data fim da hospedagem;
   3. Número de quartos;
   4. CPF do responsável pela reserva;
4. Sistema verifica se o hóspede está cadastrado; [A1]
5. Sistema verifica a disponibilidade de quartos de acordo com os dados informados no passo 3; [A2]
6. Sistema informa o valor da reserva;
7. Sistema reserva o(s) quarto(s) para o hóspede;
8. Sistema informa que a reserva foi concluída com sucesso;
9. O caso de uso se encerra.

**Fluxo alternativo:**

* [A1] Hóspede não cadastrado:
  1. O sistema informa que o hóspede não está cadastrado;
  2. O caso de uso “CSU03 – Manter Hóspede” se inicia;
  3. O caso de uso retorna para o passo 5 do Fluxo Principal.
* [A2] Não possui disponibilidade de quartos:
  1. O sistema informa que não há quartos disponíveis para serem reservados;
  2. O caso de uso retorna para o passo 2 do Fluxo Principal.

**Fluxo de Exceção:**

* [E1] Ator não informa todos os dados:
  1. O sistema informa os campos que não forma preenchidos;
  2. O caso de uso retorna para o passo 2 do Fluxo Principal.

**Pós-condição:** O ator possui um (ou mais) reserva(s).

**3 – Especifique o CSU2 apresentando os fluxos (cenários) principal, alternativo e de exceção. Utilizar um template que seja inteligível. Os casos de uso <<extend>> e <<include>> (se houver) podem ser especificados junto com o caso de uso principal.**

CSU02 – Alugar Quarto

**Sumário:** Hóspede usa o sistema para alugar um quarto

**Ator primário:** Hóspede

**Requisitos Funcionais:** RF01

**Fluxo principal:**

1. Sistema exibe a tela de Alugar de Quartos;
2. Sistema solicita a entrada de dados;
3. O ator informa os seguintes dados para o sistema: [E1]
   1. Data de início da hospedagem;
   2. Data fim da hospedagem;
   3. Número de quartos;
   4. CPF do responsável pela reserva;
4. Sistema verifica se o hóspede está cadastrado; [A1]
5. Sistema verifica a disponibilidade de quartos de acordo com os dados informados no passo 3; [A2]
6. Sistema informa o valor da locação;
7. Sistema aluga o(s) quarto(s) para o hóspede;
8. Sistema informa que a locação foi concluída com sucesso;
9. O caso de uso se encerra.

**Fluxo alternativo:**

* [A1] Hóspede não cadastrado:
  1. O sistema informa que o hóspede não está cadastrado;
  2. O caso de uso “CSU03 – Manter Hóspede” se inicia;
  3. O caso de uso retorna para o passo 5 do Fluxo Principal.
* [A2] Não possui disponibilidade de quartos:
  1. O sistema informa que não há quartos disponíveis para serem reservados;
  2. O caso de uso retorna para o passo 2 do Fluxo Principal.

**Fluxo de Exceção:**

* [E1] Ator não informa todos os dados:
  1. O sistema informa os campos que não forma preenchidos;
  2. O caso de uso retorna para o passo 2 do Fluxo Principal.

**Pós-condição:** O ator possui um (ou mais) quarto(s) alugado(s).

**4 – Especifique o CSU3 apresentando os fluxos (cenários) principal, alternativo e de exceção. Utilizar um template que seja inteligível. Os casos de uso <<extend>> e <<include>> (se houver) podem ser especificados junto com o caso de uso principal.**

CSU03 – Manter Hóspede

**Sumário:** Funcionário cria um novo registro de um hóspede

**Ator primário:** Funcionário

**Requisitos Funcionais:** RF02

**Fluxo principal:**

1. Sistema exibe a tela de Manter Hóspede;
2. Sistema solicita a entrada de dados;
3. O ator informa os seguintes dados para o sistema: [E1]
   1. CPF do hóspede;
4. Sistema verifica se o hóspede está cadastrado; [A1]
5. Sistema solicita a entrada de dados;
6. O ator informa os seguintes dados do hóspede para o sistema: [E2]
   1. Nome;
   2. Data de nascimento;
   3. E-mail;
   4. Informações do cartão de crédito;
7. O sistema armazena as informações do hóspede;
8. O caso de uso se encerra.

**Fluxo alternativo:**

* [A1] Hóspede já cadastrado:
  1. O sistema informa que o hóspede não está cadastrado;
  2. O sistema solicita a entrada de dados:
  3. O ator informa os seguintes dados do hóspede para o sistema:
     + Nome;
     + Data de nascimento;
     + E-mail;
     + Informações do cartão de crédito;
  4. O sistema atualiza as informações do hóspede;
  5. O caso de uso se encerra.

**Fluxo de Exceção:**

* [E1] Ator não informa todos os dados:
  1. O sistema informa os campos que não forma preenchidos;
  2. O caso de uso retorna para o passo 2 do Fluxo Principal.
* [E2] Ator não informa todos os dados:
  1. O sistema informa os campos que não forma preenchidos;
  2. O caso de uso retorna para o passo 5 do Fluxo Principal.

**Pós-condição:** O hóspede possui seus dados atualizados no sistema.

**5 – Especifique o CSU4 apresentando os fluxos (cenários) principal, alternativo e de exceção. Utilizar um template que seja inteligível. Os casos de uso <<extend>> e <<include>> (se houver) podem ser especificados junto com o caso de uso principal.**

CSU04 – Solicitar Serviço

**Sumário:** Funcionário mantem o registro de um hóspede

**Ator primário:** Funcionário

**Requisitos Funcionais:** RF03

**Fluxo principal:**

1. Sistema exibe a tela de Solicitar Serviços;
2. Ator informa que gostaria de solicitar um serviço;
3. Sistema exibe a lista de serviços que podem ser solicitados;
4. Sistema solicita a entrada de dados:
5. Ator informa os seguintes dados para o sistema: [A1] [E1]
   1. Serviço requerido;
   2. Quantidade do serviço requerido;
6. Sistema informa o valor do serviço solicitado;
7. Sistema registra que o serviço foi solicitado;
8. Sistema solicita se o ator gostaria de solicitar outro serviço; [A2]
9. O caso de uso se encerra.

**Fluxo alternativo:**

* [A1] Hóspede solicita um serviço indisponível:
  1. Sistema informa que o serviço solicitado está indisponível;
  2. O caso de uso retorna para o passo 2 do Fluxo Principal.
* [A2] Hóspede solicita outro serviço:
  1. O caso de uso retorna para o passo 2 do Fluxo Principal.

**Fluxo de Exceção:**

* [E1] Ator não informa todos os dados:
  1. O sistema informa os campos que não forma preenchidos;
  2. O caso de uso retorna para o passo 2 do Fluxo Principal.

**Pós-condição:** Hóspede tem seu(s) serviço(s) cadastrado(s)

**6 – Especifique o CSU5 apresentando os fluxos (cenários) principal, alternativo e de exceção. Utilizar um template que seja inteligível. O caso de uso <<extend>> e <<include>> (se houver) podem ser especificados junto com o caso de uso principal.**

CSU05 – Quitar Diárias

**Sumário:** Funcionário quita a diária de um hóspede

**Ator primário:** Funcionário

**Requisitos Funcionais:** RF05 e RF06

**Fluxo principal:**

1. Sistema exibe a tela de Quitação de diárias;
2. Sistema solicita entrada de dados:
3. Ator informa os seguintes dados para o sistema: [A1] [E1]
   1. CPF do hóspede;
4. Sistema verifica o valor das diárias;
5. Sistema verifica se o hóspede solicitou algum serviço; [A2]
6. Sistema verifica se o hóspede consumiu algum produto; [A3]
7. Sistema exibe o valor total (diárias + serviços + produtos);
8. Sistema solicita entrada de dados:
9. Ator informa os seguintes dados para o sistema: [E2]
   1. Forma de pagamento; [A4]
10. Hóspede efetua o pagamento;
11. Sistema registra que o pagamento foi efetuado;
12. O caso de uso se encerra.

**Fluxo alternativo:**

* [A1] Hóspede não possui diárias para quitar:
  1. Sistema informa que não dá diárias para quitar;
  2. O caso de uso retorna para o passo 2 do Fluxo Principal.
* [A2] Hóspede solicitou um serviço:
  1. O caso de uso “CSU07 – Calcular Serviços” se inicia;
  2. O caso de uso retorna para o passo 6 do Fluxo Principal.
* [A3] Hóspede consumiu um produto:
  1. O caso de uso “CSU08 – Calcular Consumo” se inicia;
  2. O caso de uso retorna para o passo 7 do Fluxo Principal.
* [A4] Hóspede decide efetuar pagamento utilizando cartão
  1. Sistema realiza a chamada para o ator “Sistema de Cartão”, que efetuará validações dos dados do cartão e efetuar a transação;
  2. O caso de uso retorna para o passo 10 do Fluxo Principal.

**Fluxo de Exceção:**

* [E1] Ator não informa todos os dados:
  1. O sistema informa os campos que não forma preenchidos;
  2. O caso de uso retorna para o passo 2 do Fluxo Principal.
* [E2] Ator não informa todos os dados:
  1. O sistema informa os campos que não forma preenchidos;
  2. O caso de uso retorna para o passo 8 do Fluxo Principal.

**Pós-condição:** O hóspede possui suas diárias, serviços e consumos quitados.

CSU07 – Calcular Serviços

**Sumário:** Funcionário verifica serviços do Hóspede

**Ator primário:** Funcionário

**Requisitos Funcionais:** RF05 e RF06

**Fluxo principal:**

1. Sistema calcula o valor dos Serviços.

**Fluxo alternativo:**

* N/A

**Fluxo de Exceção:**

* N/A

**Pós-condição:** Sistema calcula o valor dos serviços solicitados.

CSU08 – Calcular Consumos

**Sumário:** Funcionário verifica consumos do Hóspede

**Ator primário:** Funcionário

**Requisitos Funcionais:** RF05 e RF06

**Fluxo principal:**

1. Sistema calcula o valor dos Serviços.

**Fluxo alternativo:**

* N/A

**Fluxo de Exceção:**

* N/A

**Pós-condição:** Sistema calcula o valor dos itens consumidos.

**7 – Especifique o CSU6 apresentando os fluxos (cenários) principal, alternativo e de exceção. Utilizar um template que seja inteligível. Os casos de uso <<extend>> e <<include>> (se houver) podem ser especificados junto com o caso de uso principal.**

CSU06 – Encerrar estadia

**Sumário:** Funcionário encerra a estadia de um hóspede

**Ator primário:** Funcionário

**Requisitos Funcionais:** RF07

**Fluxo principal:**

1. Sistema exibe a tela de Encerrar Estadia;
2. O caso de uso “CSU05 – Quitar Diárias” é iniciado no passo 2 do Fluxo Principal
3. O hóspede tem sua estadia encerrada
4. O caso de uso se encerra.

**Fluxo alternativo:**

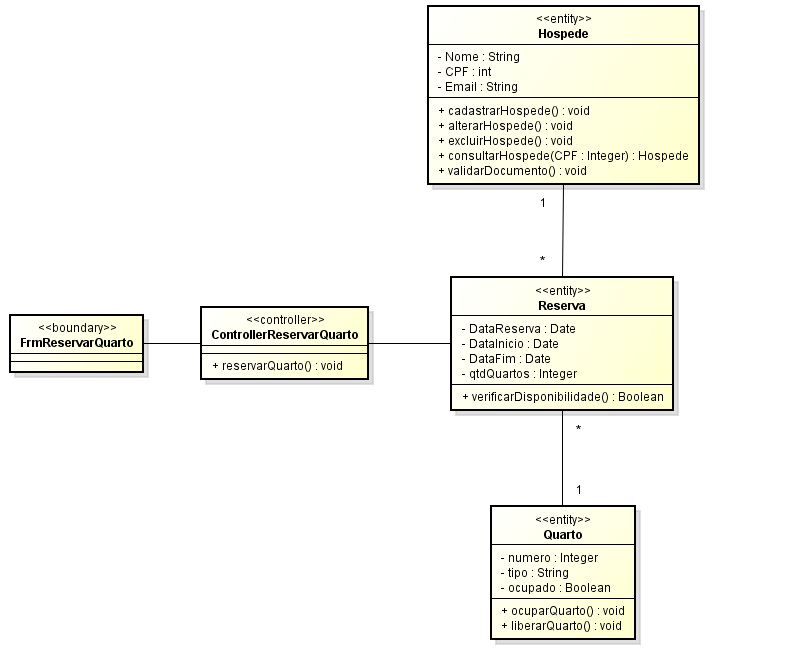
* N/A

**Fluxo de Exceção:**

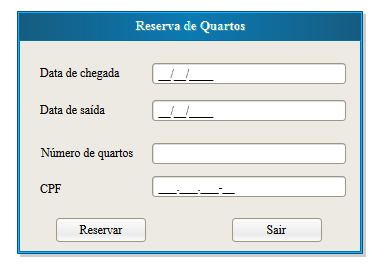
* N/A

**Pós-condição:** O hóspede possui suas estadia encerrada.

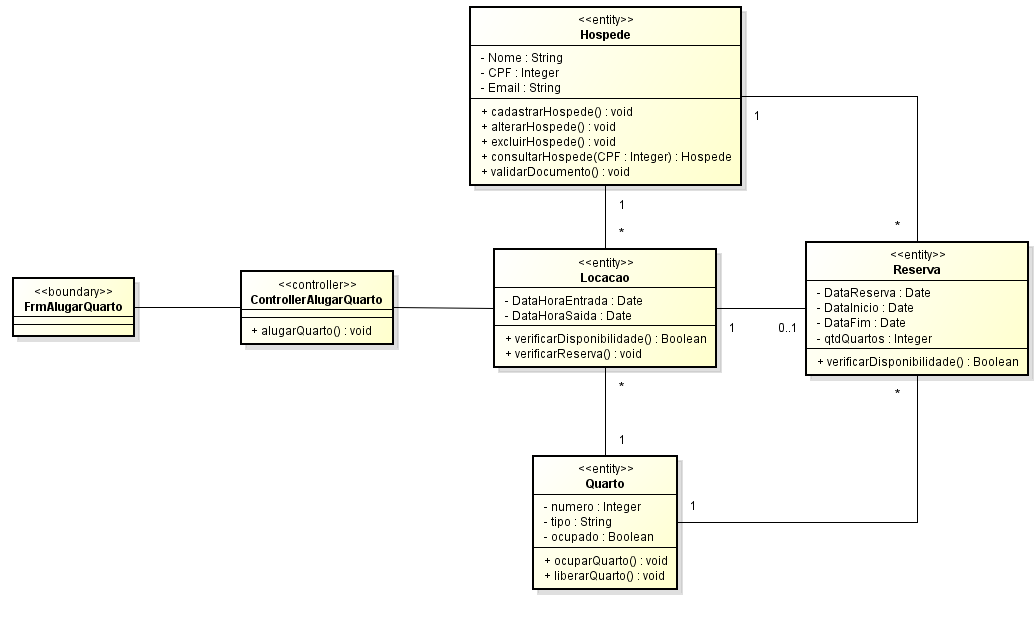
**8 – Modele uma VCP para o caso de uso utilizando a categorização BCE para o CSU01. A classe de controle deve apresentar um método no mínimo e as classes de entidade devem apresentar seus devidos atributos e um método no mínimo.**



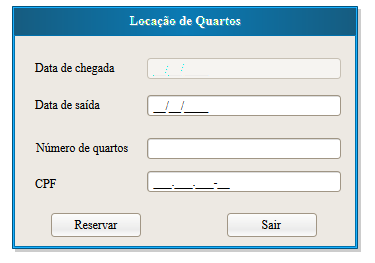
**9 – Faça o protótipo de interface de usuário para a classe <<boundary>> do CSU01.**



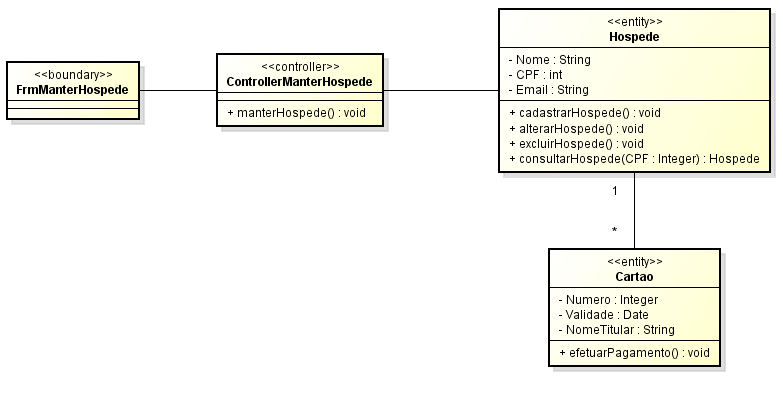
**10 – Modele uma VCP para o caso de uso utilizando a categorização BCE para o CSU02. A classe de controle deve apresentar um método no mínimo e as classes de entidade devem apresentar seus devidos atributos e um método no mínimo.**



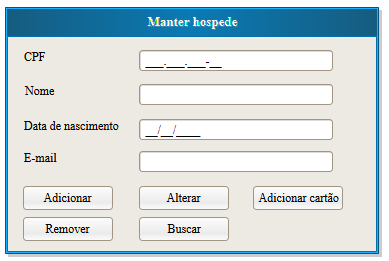
**11 – Faça o protótipo de interface de usuário para a classe <<boundary>> do CSU02.**



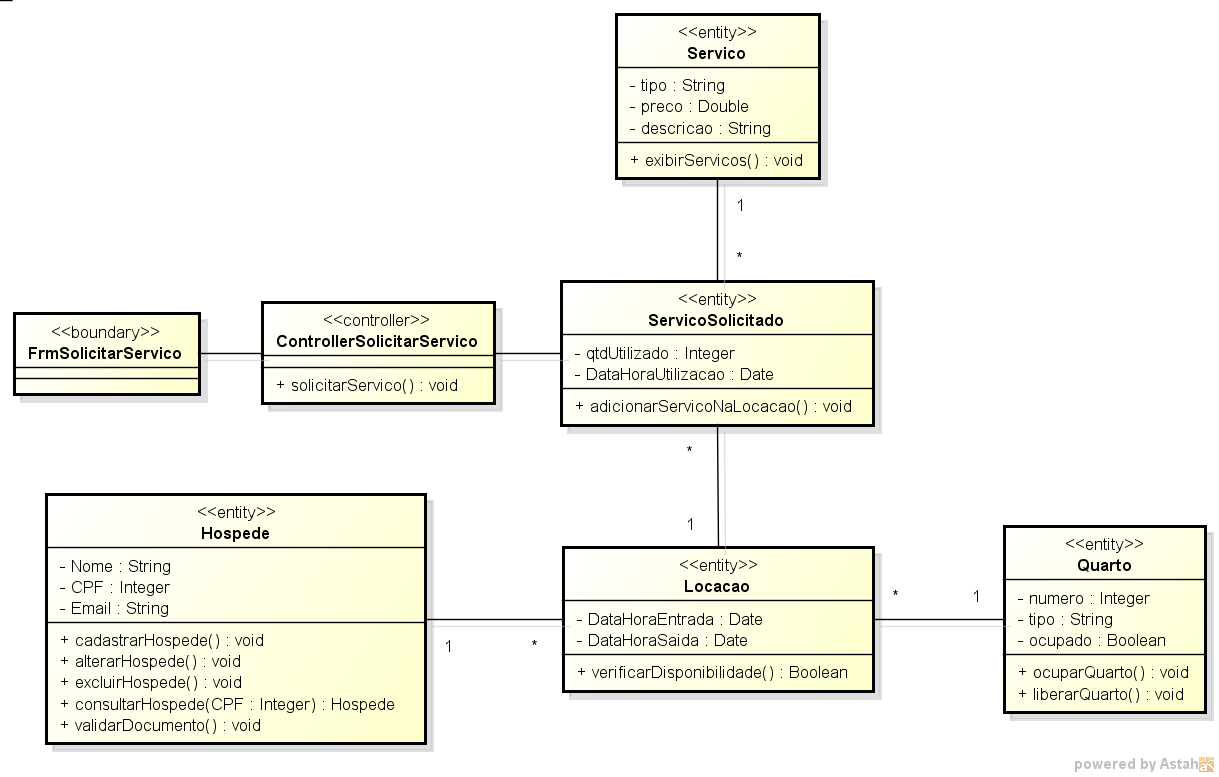
**12 – Modele uma VCP para o caso de uso utilizando a categorização BCE para o CSU3. A classe de controle deve apresentar um método no mínimo e as classes de entidade devem apresentar seus devidos atributos e um método no mínimo.**



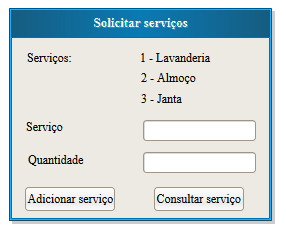
**13 – Faça o protótipo de interface de usuário para a classe <<boundary>> do CSU03.**



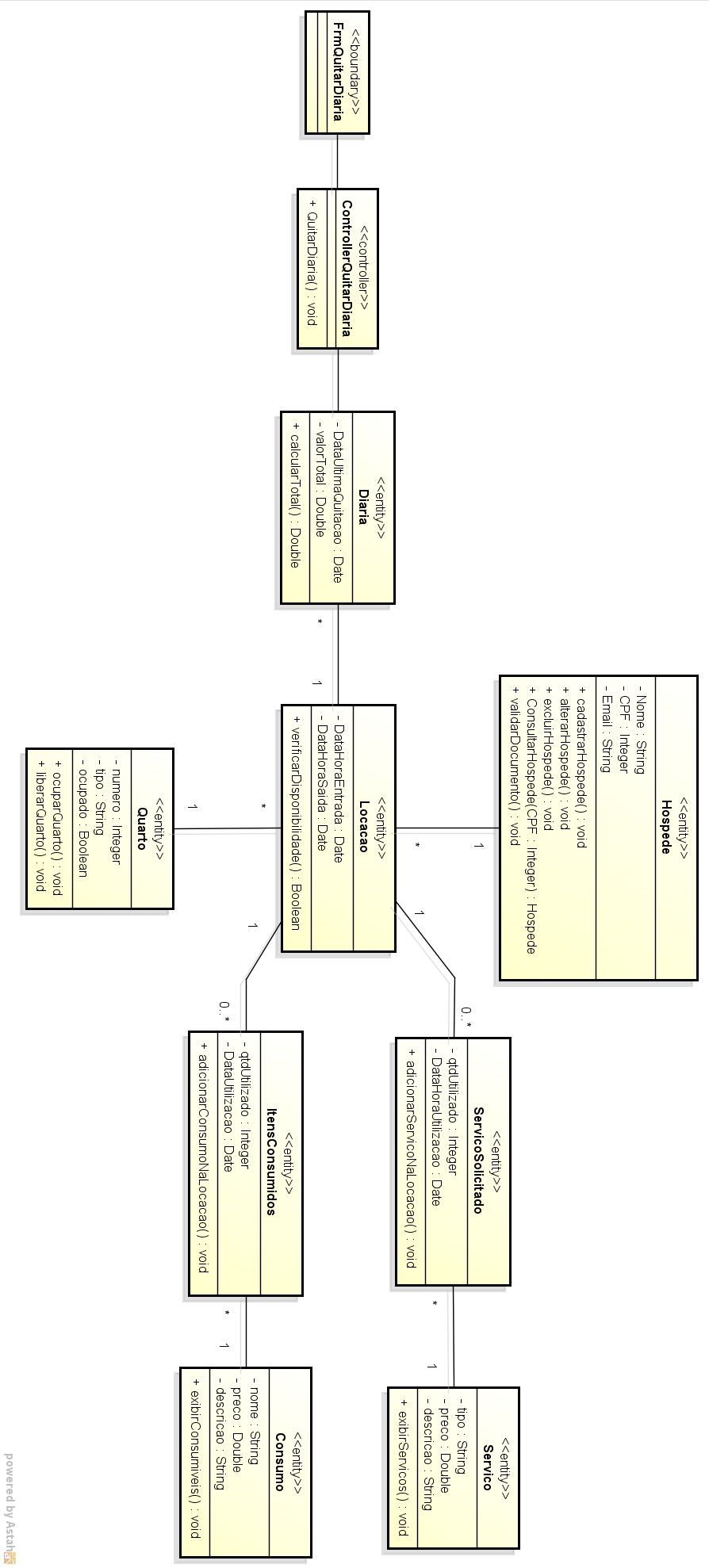
**14 – Modele uma VCP para o caso de uso utilizando a categorização BCE para o CSU4. A classe de controle deve apresentar um método no mínimo e as classes de entidade devem apresentar seus devidos atributos e um método no mínimo.**



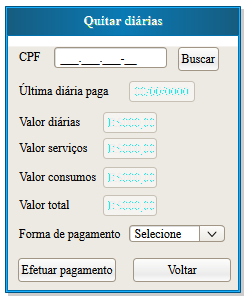
**15 – Faça o protótipo de interface de usuário para a classe <<boundary>> do CSU04.**



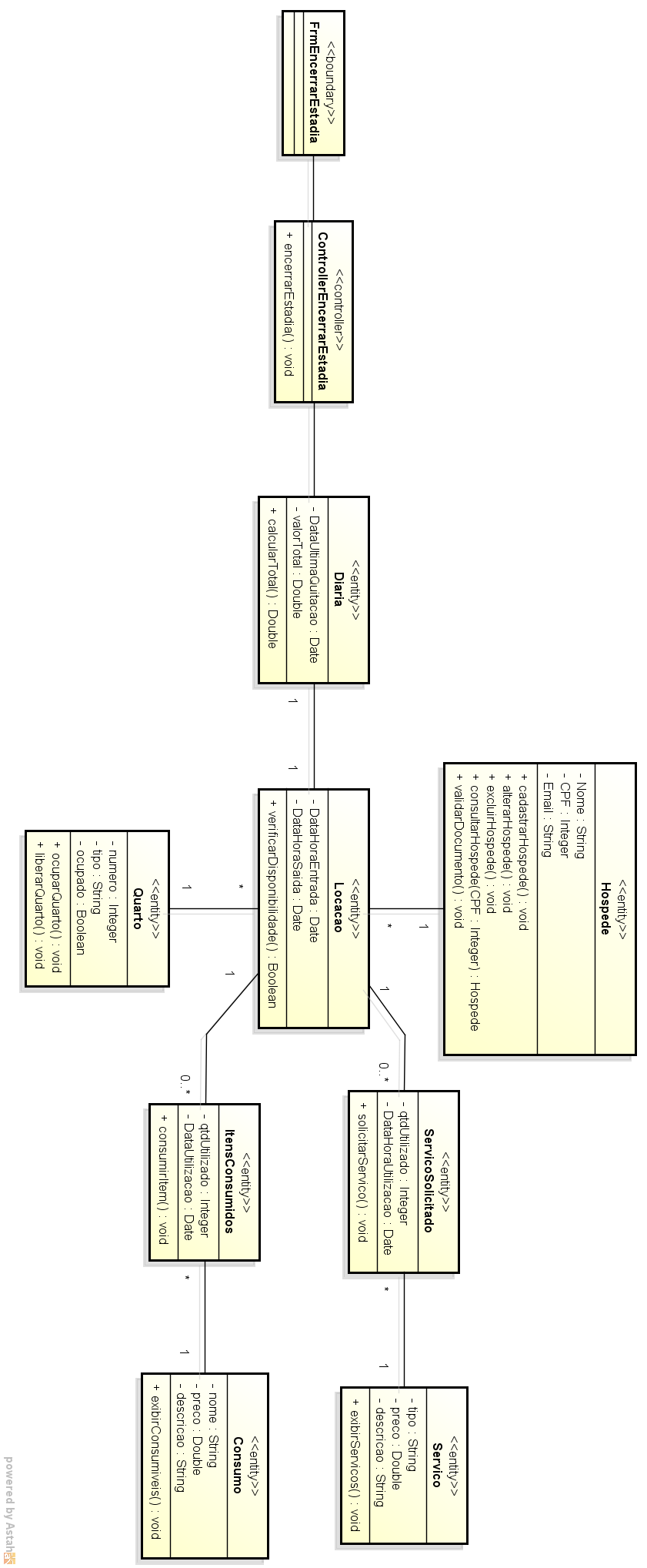
**16 – Modele uma VCP para o caso de uso utilizando a categorização BCE para o CSU5. A classe de controle deve apresentar um método no mínimo e as classes de entidade devem apresentar seus devidos atributos e um método no mínimo.**



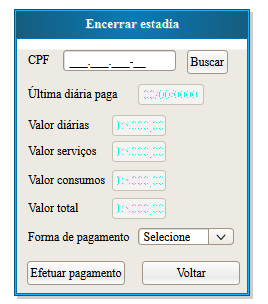
**17- Faça o protótipo de interface de usuário para a classe <<boundary>> do CSU05.**



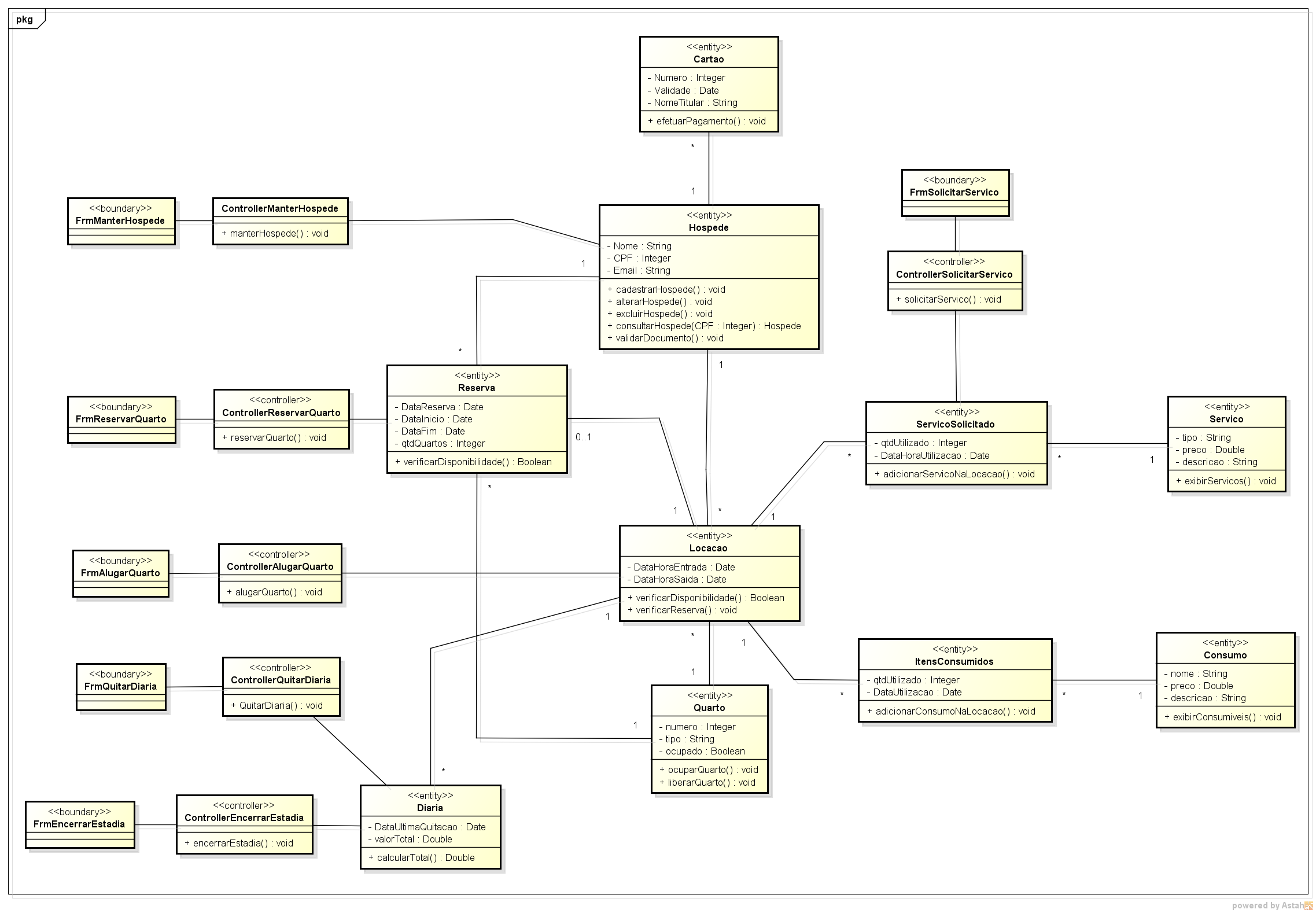
**18 – Modele uma VCP para o caso de uso utilizando a categorização BCE para o CSU6. A classe de controle deve apresentar um método no mínimo e as classes de entidade devem apresentar seus devidos atributos e um método no mínimo.**



**19 – Faça o protótipo de interface de usuário para a classe <<boundary>> do CSU06.**

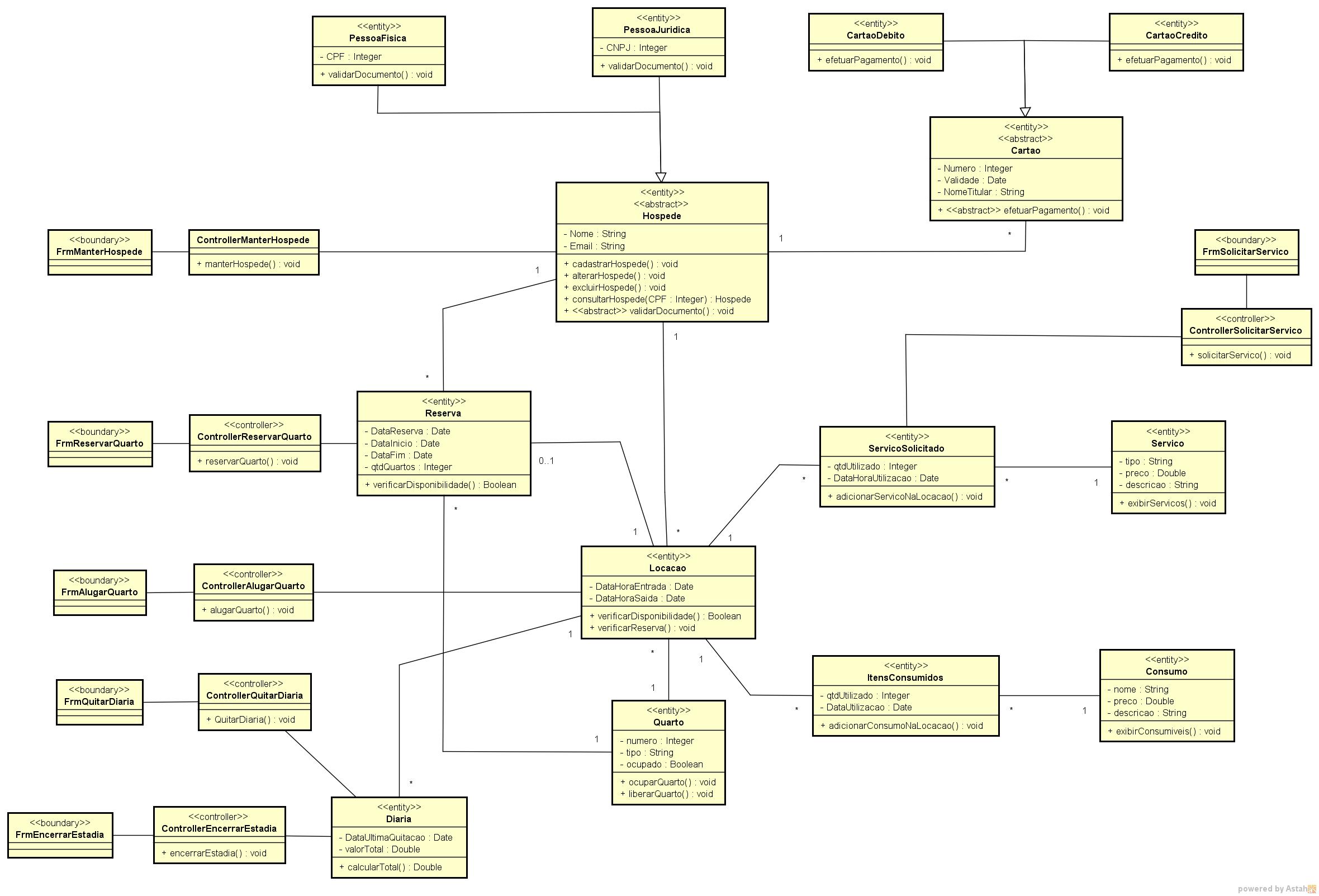


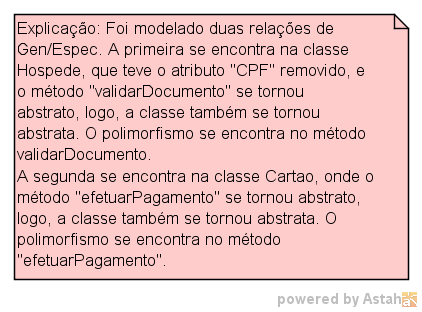
**20 – Modele um diagrama de classes de projeto a partir das VCPs modeladas e mantenha a utilização da categorização BCE. Os devidos atributos e métodos devem continuar sendo exibidos.**



**21 – Modele duas relações de gen/espec e ative o princípio de polimorfismo universal de inclusão em cada uma delas.**

**Justifique a razão de existência de cada gen/espec e das operações polimórficas.**





**22 – As relações de gen/espec violam o Princípio de Liskov? Justifique a tua resposta.**

Ambas as gen/spec modeladas não violam o princípio de Liskov, pois as especificações podem ser substituídas pela generalização sem que seus atributos, métodos e relacionamentos percam o sentido.

**23 – Apresente a estrutura básica de código em JAVA, C# ou C++ para implementar as relações de gen/espec e as operações polimórficas.**

**public** **abstract** **class** Hospede {

**private** String Nome;

**private** String Email;

**public** **void** cadastrarHospede(){

}

**public** **void** alterarHospede(){

}

**public** **void** excluirHospede(){

}

**public** Hospede consultarHospede(**int** CPF){

**return** **null**;

}

**public** **abstract** **void** validarDocumento();

**public** String getNome() {

**return** Nome;

}

**public** **void** setNome(String nome) {

Nome = nome;

}

**public** String getEmail() {

**return** Email;

}

**public** **void** setEmail(String email) {

Email = email;

}

}

**public** **class** PessoaFisica **extends** Hospede {

**private** **int** cpf;

@Override

**public** **void** validarDocumento() {

System.***out***.println("Validando CPF");

}

**public** **int** getCpf() {

**return** cpf;

}

**public** **void** setCpf(**int** cpf) {

**this**.cpf = cpf;

}

}

**public** **class** PessoaJuridica **extends** Hospede {

**private** **int** cnpj;

@Override

**public** **void** validarDocumento() {

System.***out***.println("Validando CNPJ");

}

**public** **int** getCnpj() {

**return** cnpj;

}

**public** **void** setCnpj(**int** cnpj) {

**this**.cnpj = cnpj;

}

}

**import** java.util.Date;

**public** **abstract** **class** Cartao {

**private** **int** numero;

**private** Date validade;

**private** String nomeTitular;

**public** **abstract** **void** efetuarPagamento();

**public** **int** getNumero() {

**return** numero;

}

**public** **void** setNumero(**int** numero) {

**this**.numero = numero;

}

**public** Date getValidade() {

**return** validade;

}

**public** **void** setValidade(Date validade) {

**this**.validade = validade;

}

**public** String getNomeTitular() {

**return** nomeTitular;

}

**public** **void** setNomeTitular(String nomeTitular) {

**this**.nomeTitular = nomeTitular;

}

}

**public** **class** CartaoDebito **extends** Cartao {

@Override

**public** **void** efetuarPagamento() {

System.***out***.println("Efetuando pagamento com débito");

}

}

**public** **class** CartaoCredito **extends** Cartao{

@Override

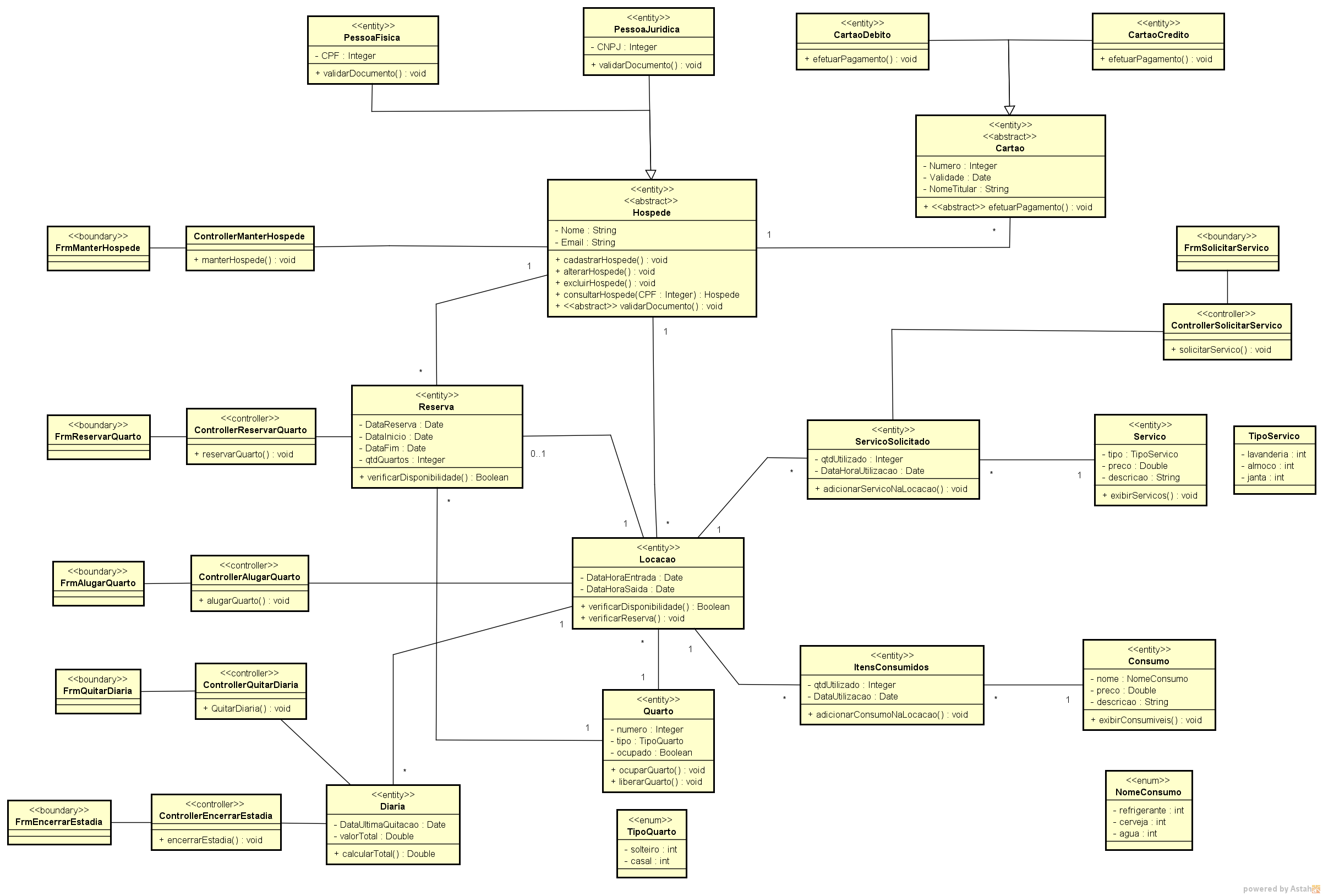
**public** **void** efetuarPagamento() {

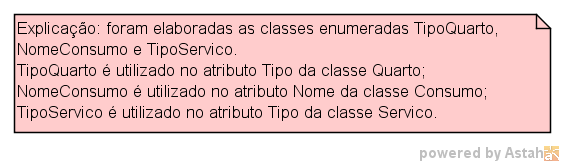
System.***out***.println("Efetuando pagamento com crédito");

}

}

**24 – Modele três classes enumeradas e utilize as mesmas como tipos de atributos. Justifique a existência de cada uma das classes enumeradas modeladas.**





**25 – Modele seis membros estáticos, sendo três atributos e três métodos. Justifique a criação de existência de cada um dos membros estáticos modelados.**

