

Engenharia de Software 3

Lista I

Andrei Lucas 1680481521025
Daniel Augusto 1680481521030
Diego Melo 1680481521036
Elias Sanai 1680481521016
Henrique Fernandes 141682263
Willian Messias 1680481521034

Analise os Requisitos Funcionais de um Projeto de Desenvolvimento de Software de Museu apresentados logo

abaixo e realize os exercícios na sequência.

RF01: Gestão de Visita

Este requisito deve gerenciar o fluxo de visitantes do museu por meio de um cartão magnético fornecido para o visitante na entrada, sendo possível armazenar os dados básicos de cada visitante para envio de convite de exposição ou evento.

RF02: Gestão de Acervo

Este requisito deve gerenciar as obras presentes no museu por meio de um cadastro unificado de acervo físico (obras em geral) e virtual (Documentos Históricos), onde poderão ser realizadas consultas das obras.

RF03: Gestão de Exposição

Este requisito deve gerenciar as exposições, temporárias ou permanentes do museu, permitindo listar as obras exibidas em cada exposição, além de mostrar a salas reservadas e o máximo de visitantes.

RF04: Gestão de Restauração

Este requisito deve gerenciar o processo de restauração de obras, desde a abertura da solicitação de serviço, duração e andamento até a conclusão da restauração da obra.

RF05: Gestão de Evento

Este requisito deve gerenciar as informações do evento a ser realizado, assim como quem será o responsável e o número de visitantes para o evento em questão.

RF06: Gestão de Venda de Souvenir

Este requisito deve gerenciar a loja de souvenirs, desde o seu estoque até a aquisição de novos produtos para a loja. Fornece relatórios de vendas para o gerente deste setor.

Regras de Negócio

RN01 - Pagamento na entrada

O visitante deve pagar uma taxa de entrada no valor de 10,00 e 5,00 para meia entrada
Se o visitante estiver cadastrado ele possui um desconto de 15% na entrada

RN02 - Capacidade Máxima de Exposição

O cartão magnético deve ser passado no validador eletrônico no momento da entrada do visitante na sala de sessão para controle do fluxo de visitantes.

RN03 - Definição de tipo de Exposição

O tipo de uma exposição se ela é permanente ou temporária deve ser definido pelo atributo na exposição data de termino onde se o atributo não possuir valor a exposição é permanente e se ela possuir uma data ela é temporária

1- Especifique textualmente um caso de uso (CSU01) para o RF01, apresentando os fluxos (cenários) principal, alternativo e de exceção. Utilize um template que seja inteligível. Possíveis casos de uso <<extend>> ou <<include>> podem ser especificados junto com o caso de uso principal.

Exercício 01

Gerenciar Visitas (CSU01)

Ator Primário: Atendente

Ator Secundário: Visitante

Précondição: O atendente está identificado pelo sistema

Fluxo Principal:

- 1 O Visitante solicita a entrada no museu
- 2 O Atendente verifica cadastro do Visitante
- 3 O Atendente indica a taxa a pagar ao Visitante (RN01)
- 4 O Atendente prepara o cartão magnético e entrega ao Visitante

Fluxos Alternativo (2): Visitante não cadastrado

- a. Se o visitante não possui cadastro no sistema: O atendente reporta a possibilidade de realizar cadastro
- b. Se o visitante aceitar fazer cadastro: O sistema apresenta um formulário padrão para que os detalhes do visitante (nome, cpf, data nascimento, celular, email) sejam incluídos
- c. O sistema verifica a validade dos dados. Se os dados forem válidos, inclui o novo visitante; caso contrário, o sistema reporta o fato, solicita novos dados e repete a verificação
- d. O caso de uso retorna ao passo 2

Fluxo Alternativo (2): Visitante cadastrado

- a. Se o Visitante possui cadastro no sistema: O Atendente requisita ao Visitante seu CPF

Fluxos de Exceção (3): Visitante não realiza pagamento da taxa

- a. Se o Visitante não realiza o pagamento da taxa: O caso de uso se encerra

Póscondição: Visitante recebe permissão para entrar no museu

Regras de Negócio: RN01

2- Especifique textualmente um caso de uso (CSU02) para o RF02, apresentando os fluxos (cenários) principal, alternativo e de exceção. Utilize um template que seja inteligível. Possíveis casos de uso <>extend>> ou <>include>> podem ser especificados junto com o caso de uso principal.

Manter Acervo (CSU02)

Ator Primário: Gerente

Précondição: O Gerente está identificado pelo sistema

Fluxo Principal:

1. Gerente requisita a manutenção do acervo
2. O sistema apresenta as operações que podem ser realizadas para as obras físicas e virtuais: Inclusão, Exclusão, Alteração ou Consulta
3. O Gerente seleciona a operação desejada: Inclusão, Exclusão, Alteração, Consulta ou opta por finalizar o caso de uso.
4. Se o Gerente deseja continuar com a manutenção, o caso de uso retorna ao passo 2; caso contrário, o caso de uso termina.

Fluxo Alternativo (3): Inclusão

- a. O Gerente requisita a inclusão de uma obra física ou virtual.
- b. O sistema apresenta um formulário padrão para que os detalhes da obra (código da obra , tipo de obra, autor, dados biográficos, título, data da obra, técnica, local acervo físico, status da obra) sejam incluídos.
- c. O Gerente fornece os detalhes da nova obra.
- d. O sistema verifica a validade dos dados. Se os dados forem válidos, inclui a nova obra; caso contrário, o sistema reporta o fato, solicita novos dados e repete a verificação.

Fluxo Alternativo (3): Remoção

- a. O Gerente seleciona uma obra e requisita o sistema que a remova.
- b. Se a obra pode ser removida, o sistema realiza a remoção; caso contrário, o sistema reporta o fato.

Fluxo Alternativo (3) : Alteração

- a. O Gerente altera um ou mais dos detalhes sobre uma obra e requisita a sua atualização.

- b. O sistema verifica a validade dos dados e, se eles forem válidos, altera os dados na lista de obras do museu.

Fluxo Alternativo (3): Consulta

- a. O Gerente solicita a realização de uma consulta sobre a lista de obra.
- b. O sistema apresenta uma lista com os códigos de todas as obras, permitindo que o usuário selecione a obra desejado.
- c. O Gerente seleciona uma obra.
- d. O sistema apresenta os detalhes da obra.

Pós-condições: uma obra foi inserida ou removida, ou seus detalhes foram alterados.

3- Especifique textualmente um caso de uso (CSU03) para o RF03, apresentando os fluxos (cenários) principal, alternativo e de exceção. Utilize um template que seja inteligível. Possíveis casos de uso <<extend>> ou <<include>> podem ser especificados junto com o caso de uso principal.

Manter Exposição (CSU03)

Ator Primário: Gerente, atendente

Précondição: O Gerente/Atendente está identificado pelo sistema

Fluxo Principal:

1. Gerente/Atendente requisita a manutenção de exposições
2. O sistema apresenta as operações que podem ser realizadas para as exposições: Inclusão, Exclusão, Alteração ou Consulta
3. O Gerente/Atendente seleciona a operação desejada: Inclusão, Exclusão, Alteração, Consulta ou opta por finalizar o caso de uso.
4. Se o Gerente/Atendente deseja continuar com a manutenção, o caso de uso retorna ao passo 2; caso contrário, o caso de uso termina.

Fluxo Alternativo (3): Inclusão

- a. O Gerente/Atendente requisita a inclusão de uma exposição.
- b. O sistema apresenta um formulário padrão para que os detalhes da exposição (código, nome, data de inicio, data de termino, capacidade máxima de visitantes) sejam incluídos. (RN02, RN03)
- c. O Gerente/Atendente fornece os detalhes da nova exposição.
- d. O sistema apresenta uma lista das obras para serem selecionadas
- e. O sistema apresenta uma lista das salas para serem selecionadas

f. O sistema verifica a validade dos dados. Se os dados forem válidos e pelo menos uma sala e uma obra foi selecionada, inclui a nova exposição; caso contrário, o sistema reporta o fato, solicita novos dados e volta para o passo 3.

Fluxo Alternativo (3): Remoção

- a. O Gerente/Atendente seleciona uma exposição e requisita o sistema que a remova.
- b. Se a exposição pode ser removida, o sistema realiza a remoção; caso contrário, o sistema reporta o fato.

Fluxo Alternativo (3) : Alteração

- a. O Gerente/Atendente altera um ou mais dos detalhes sobre uma exposição e requisita a sua atualização.
- b. O sistema verifica a validade dos dados e, se eles forem válidos, altera os dados na lista de exposições do museu.

Fluxo Alternativo (3): Consulta

- a. O Gerente/Atendente solicita a realização de uma consulta sobre a lista de exposições.
- b. O sistema apresenta uma lista com os códigos de todas as exposições, permitindo que o usuário selecione a exposição desejada.
- c. O Gerente/Atendente seleciona uma exposição.
- d. O sistema apresenta os detalhes da exposição.

Pós-condições: uma exposição foi inserida ou removida, ou seus detalhes foram alterados.

Regras de Negócio: RN02, RN03

4- Especifique textualmente um caso de uso (CSU04) para o RF04, apresentando os fluxos (cenários) principal, alternativo e de exceção. Utilize um template que seja inteligível. Possíveis casos de uso <<extend>> ou <<include>> podem ser especificados junto com o caso de uso principal.

Gerenciar Restauração (CSU04)

Autor Primário: Gerente

Autor Secundários: Empresa de Restauração

Précondição: O Gerente está identificado pelo sistema

Fluxo Principal:

1. Gerente requisita gerenciar restaurações
2. O sistema apresenta as operações que podem ser realizadas para as restaurações: Abertura, Consultar andamento, Concluir em solicitações de serviço
3. O Gerente seleciona a operação desejada.
4. Retorna para o passo 2

Fluxo Alternativo (3): Abertura

- a. O Gerente requisita uma abertura de solicitação de serviço
- b. O sistema apresenta um formulário padrão para que os detalhes da solicitação de serviço (código de serviço, descrição, empresa de restauracao, data de inicio, data de previsão de termino, obra que será restaurada, valor orçado) sejam incluídos.
- c. O Gerente fornece os detalhes da nova restauração.
- d. O sistema verifica a validade dos dados. Se os dados forem válidos, inclui a nova solicitação; caso contrário, o sistema reporta o fato, solicita novos dados e repete a verificação.
- e. O sistema envia uma declaração do Gerente para a Empresa de Restauração via email

Fluxo Alternativo (3): Consultar andamento

- a. O Gerente solicita a realização de uma consulta sobre a lista de restaurações em andamento.
- b. O sistema apresenta uma lista com os códigos de todas as restaurações em andamento, permitindo que o usuário selecione a restaurações desejada.
- c. O Gerente seleciona uma restauração.
- d. O sistema apresenta os detalhes da restauração.

Fluxo Alternativo (3) : Dar baixa

- a. O Gerente solicita a realização de uma consulta sobre a lista de restaurações.
- b. O sistema apresenta uma lista com os códigos de todas as restaurações, permitindo que o usuário selecione a restaurações desejada.
- c. O Gerente seleciona uma restauração.
- d. O sistema apresenta os detalhes da restauração.
- e. O Gerente seleciona a opção dar baixa na restauração escolhida
- f. O sistema apresenta um formulário padrão para que os detalhes da baixa de serviço sejam incluídas (data de baixa, valor orçado, observação da conclusao)

Pós-condições: uma ou mais obras podem ter seus status da obra modificada entre (normal e sendo restaurada)

5- Especifique textualmente um caso de uso (CSU05) para o RF05, apresentando os fluxos (cenários) principal, alternativo e de exceção. Utilize

um template que seja inteligível. Possíveis casos de uso <> ou <> podem ser especificados junto com o caso de uso principal.

Manter Evento (CSU05)

Autor Primário: Gerente

Précondição: O Gerente está identificado pelo sistema

Fluxo Principal:

1. Gerente requisita a manutenção de eventos
2. O sistema apresenta as operações que podem ser realizadas para os eventos: Inclusão, Exclusão, Alteração, Consulta ou Finalizar caso de uso
3. O Gerente seleciona a operação desejada
4. Retorna para o passo 2

Fluxo Alternativo (3): Inclusão

- a. O Gerente requisita a inclusão de um evento.
- b. O sistema apresenta um formulário em branco para que os detalhes do evento (código, nome, descrição, data de inicio, data de termino, nome responsável, numero participantes) sejam incluídos.
- c. O Gerente fornece os detalhes do novo evento.
- d. O sistema apresenta uma lista das salas para serem selecionadas
- e. O sistema verifica a validade dos dados. Se os dados forem válidos e pelo menos uma sala foi selecionada, inclui o novo evento; caso contrário, o sistema reporta o fato, solicita novos dados e repete a verificação.

Fluxo Alternativo (3): Remoção

- a. O Gerente seleciona um evento e requisita o sistema que o remova.
- b. Se o evento pode ser removido, o sistema realiza a remoção; caso contrário, o sistema reporta o fato.

Fluxo Alternativo (3): Alteração

- a. O Gerente altera um ou mais dos detalhes sobre um evento e requisita a sua atualização.
- b. O sistema verifica a validade dos dados e, se eles forem válidos, altera os dados na lista do evento.

Fluxo Alternativo (3): Consulta

- a. O Gerente solicita a realização de uma consulta sobre a lista de evento.
- b. O sistema apresenta uma lista com os códigos de todos os eventos, permitindo que o usuário selecione um evento desejado.
- c. O Gerente seleciona um evento.

d. O sistema apresenta os detalhes do evento.

Pós-condições: um evento foi inserido ou removido, ou seus detalhes foram alterados.

6- Especifique textualmente um caso de uso (CSU06) para o RF06, apresentando os fluxos (cenários) principal, alternativo e de exceção. Utilize um template que seja inteligível. Possíveis casos de uso <<extend>> ou <<include>> podem ser especificados junto com o caso de uso principal.

Gerenciar Venda Souvenir (CSU06)

Autor Primário: Gerente, Atendente

Autor Secundário: Sistema Venda Souvenir (SVS)

Précondição: O Gerente/Atendente está identificado pelo sistema

Fluxo Principal:

1. Gerente/Atendente requisita o Gerenciamento de Venda de Souvenir
2. O sistema apresenta as operações que podem ser realizadas para a Venda de Souvenir: Movimentação de Estoque, Incluir Produto, Gerar relatório de protudo, gerar relatorio de estoque ou Finalizar caso de uso
3. O Gerente/Atendente seleciona a operação desejada
4. Retorna para o passo 2

Fluxo Alternativo (3): Movimentação de Estoque

- a. O Gerente/Atendente requisita a movimentação de estoque.
- b. O sistema apresenta uma lista dos produtos com seus códigos para serem selecionados
- c. O Gerente/Atendente seleciona um produto
- d. O sistema apresenta um formulário para definir os atributos (quantidade de estoque movimentada, tipo movimentação [entrada, saída, definição de estoque], observação)
- e. O Gerente/Atendente insere as informações nos campos
- f. O sistema verifica a validade dos dados. Se os dados forem válidos, inclui uma movimentação de estoque; caso contrário, o sistema reporta o fato, solicita novos dados e repete a verificação.
- g. O sistema envia os dados para o SVS

Fluxo Alternativo (3): Incluir Produto

- a. O Gerente/Atendente requisita a inclusão de um produto.

- b. O sistema apresenta um formulário padrão para que os detalhes do produto (código barras, descrição, valor produto, estoque atual, categoria) sejam incluídos.
- c. O Gerente/Atendente fornece os detalhes do novo produto.
- e. O sistema verifica a validade dos dados. Se os dados forem válidos, inclui o novo produto; caso contrário, o sistema reporta o fato, solicita novos dados e repete a verificação.
- f. O sistema envia os dados para o SVS

Fluxo Alternativo (3): Gerar relatório

- a. O Gerente/Atendente requisita gerar relatórios de vendas
- b. O sistema acessa os dados do SVS
- c. O sistema gera os relatórios
- d. O sistema apresenta os relatórios

Fluxos de Exceção (Movimentação de Estoque - g): Houve uma falha no envio de Movimentação de Estoque

- a. O sistema não consegue enviar os dados para o SVS
- b. O sistema reporta o fato e o retorna para Movimentação de Estoque - e

Fluxos de Exceção (Incluir Produto - f): Houve uma falha na inclusão de um produto

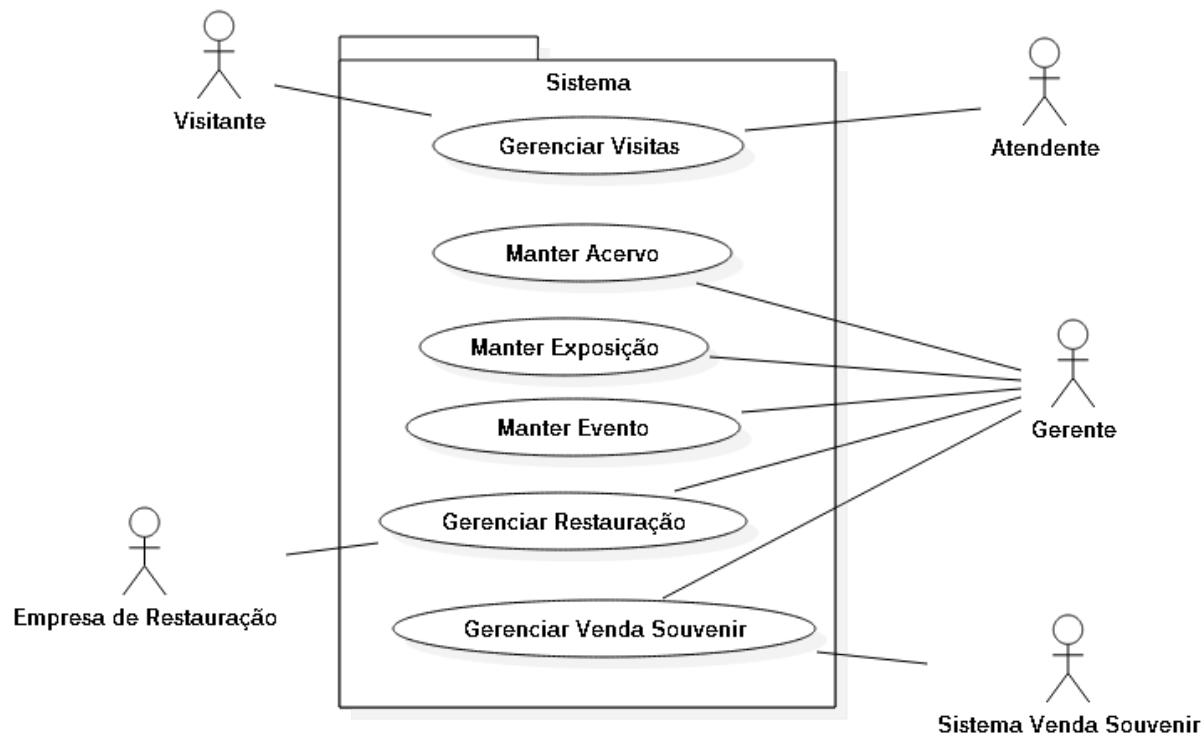
- a. O sistema não consegue enviar os dados para o SVS
- b. O sistema reporta o fato e o retorna para Incluir produto - f

Fluxos de Exceção (Gerar relatório - b): Houve uma falha na obtenção de dados

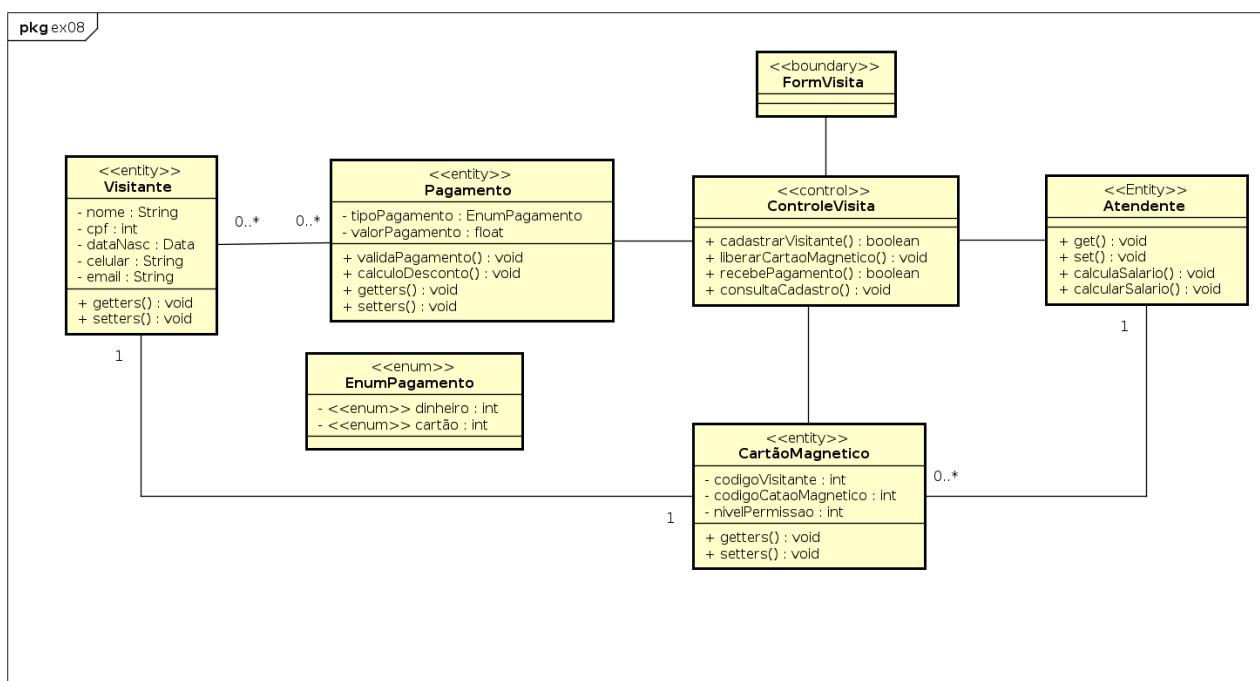
- a. O sistema não consegue obter os dados a partir do SVS
- b. Retorna para o passo 2.

Pós-condições: um estoque foi alterado ou um novo produto foi incluído.

7- Modele um Diagrama de Casos de Uso (DCU) com base nas especificações textuais.



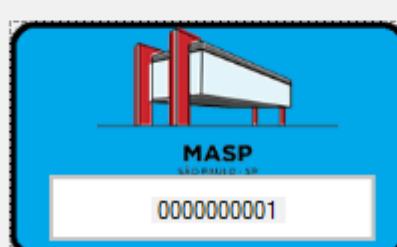
8- Modele uma VCP para o CSU01, utilizando a categorização BCE. A classe de controle deve apresentar dois métodos no mínimo e as classes de entidade devem apresentar seus devidos atributos e métodos. Faça também o protótipo de interface de usuário para a classe <<boundary>> do CSU01.



MASP - Visitas



Dados Cartão Magnético



Cod. Visitante

Cod. Cartão Magnético

Nível Permissão

VISITANTE

Cod. Visitante

Nome

CPF Data Nasc. //

Celular Email

Forma de Pagamento

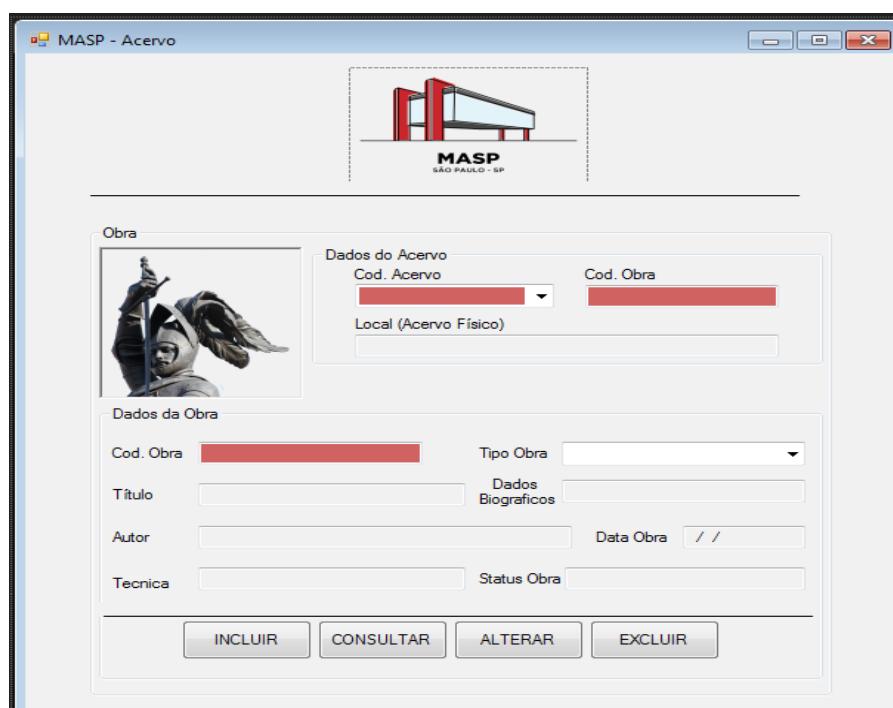
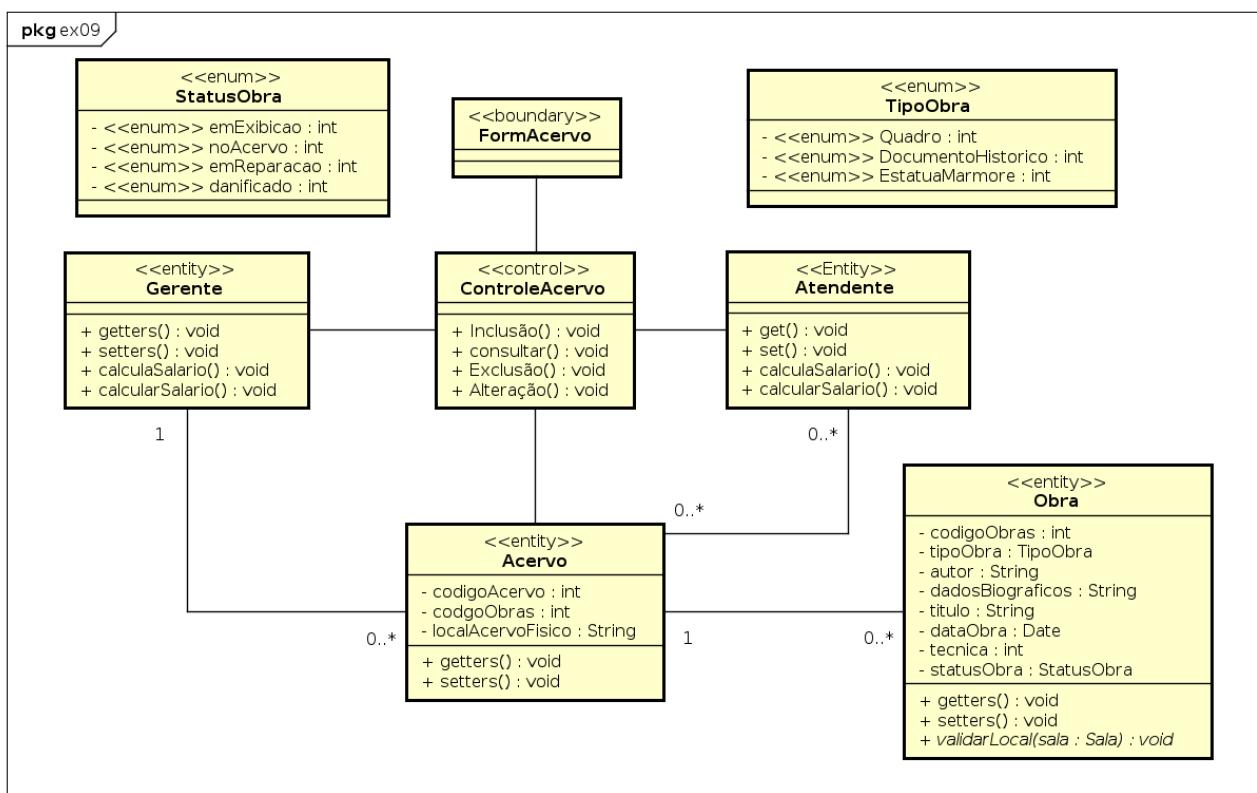
Dinheiro Cartão

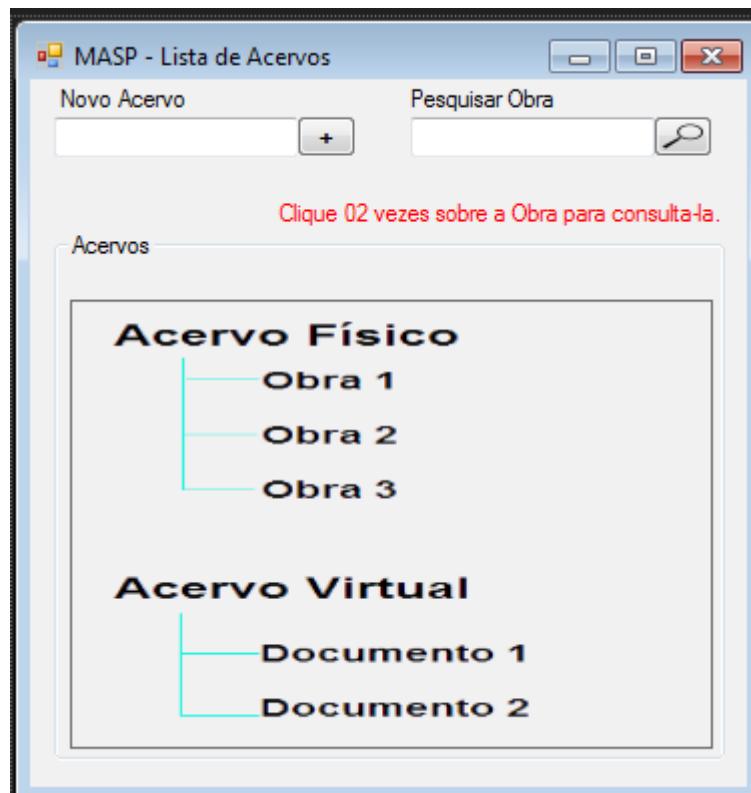
15,00

CONFIRMAR

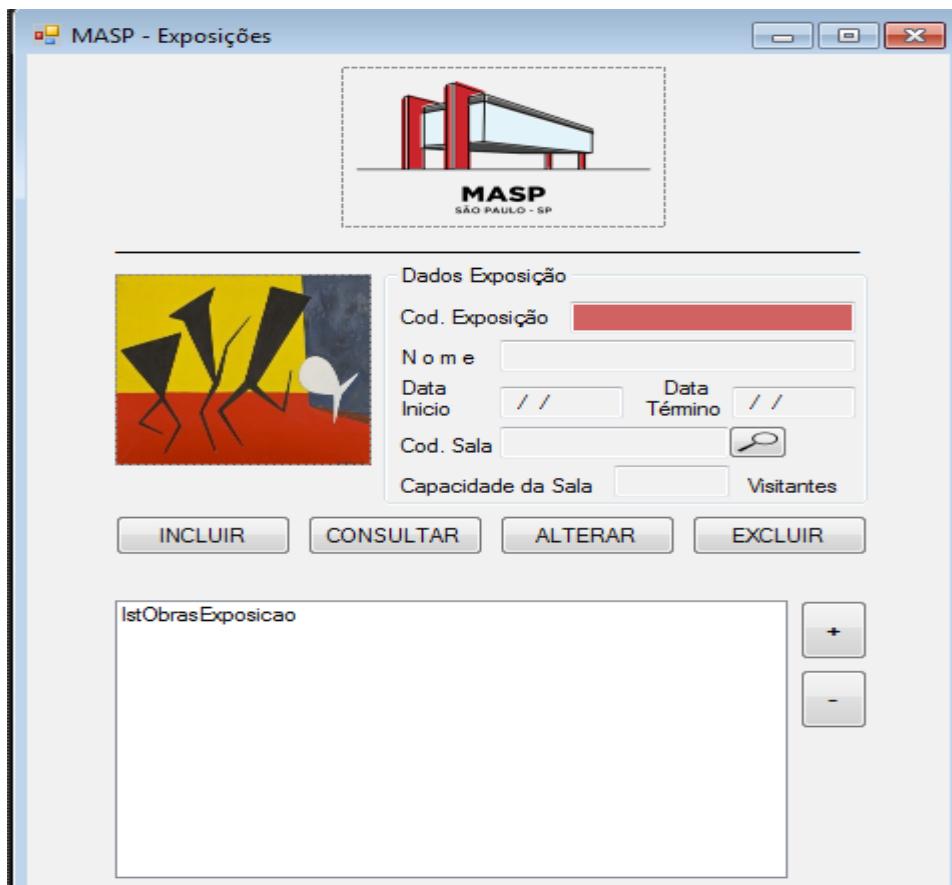
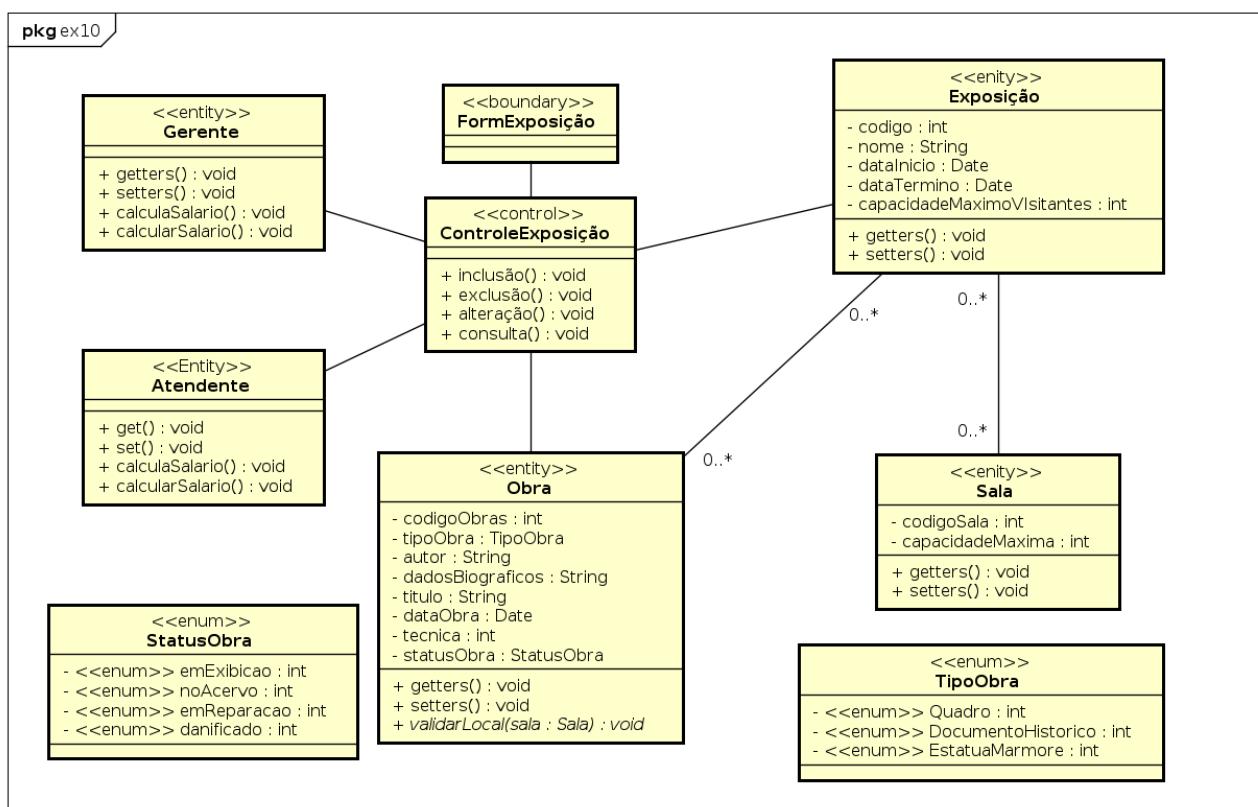
CANCELAR

9- Modele uma VCP para o CSU02, utilizando a categorização BCE. A classe de controle deve apresentar dois métodos no mínimo e as classes de entidade devem apresentar seus devidos atributos e métodos. Faça também o protótipo de interface de usuário para a classe <>boundary>> do CSU02.

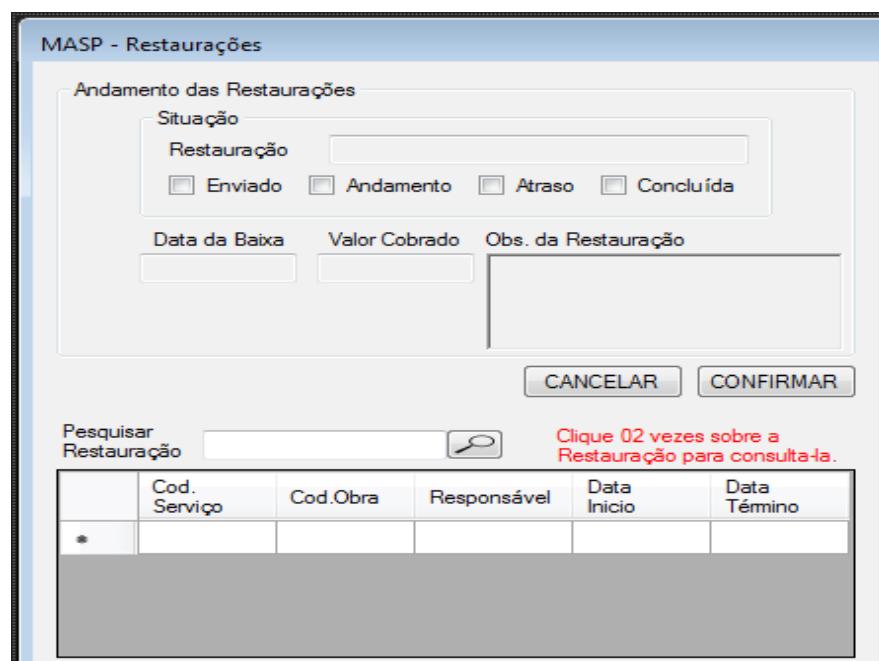
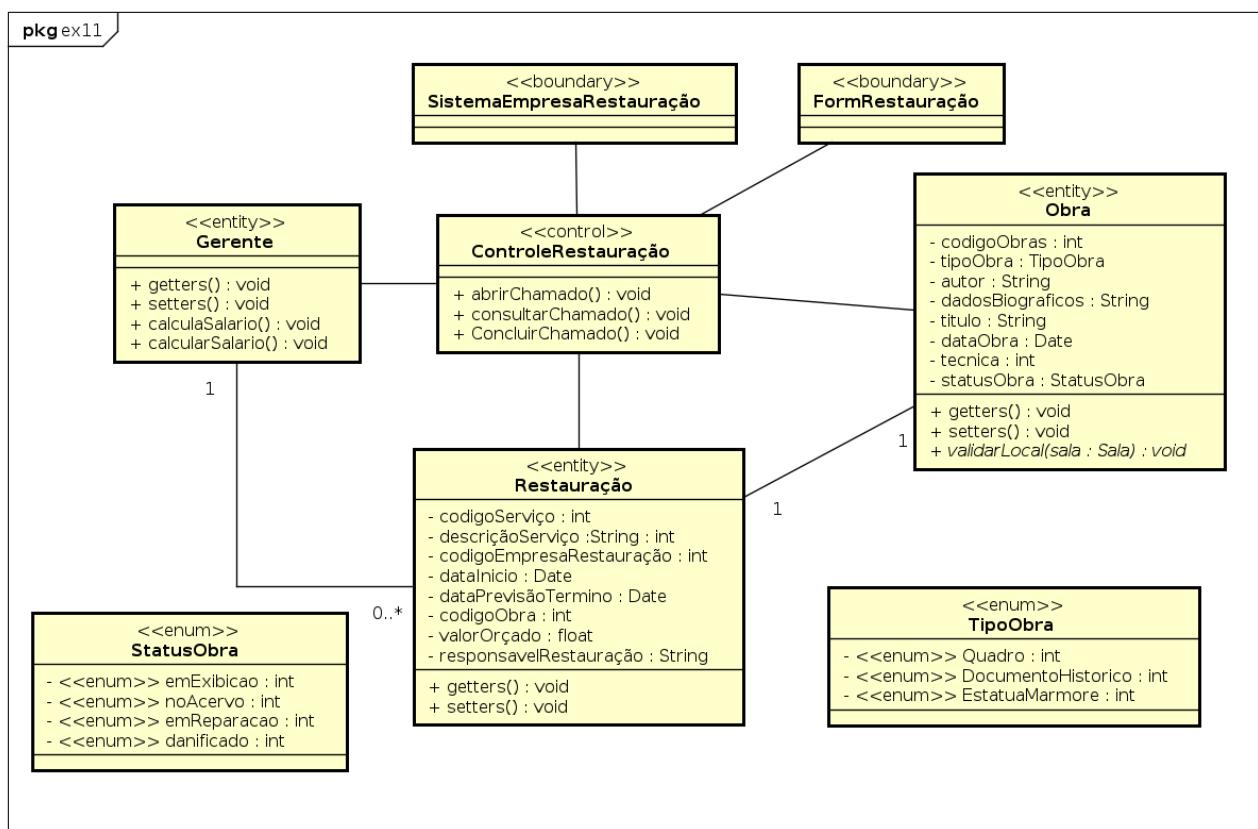


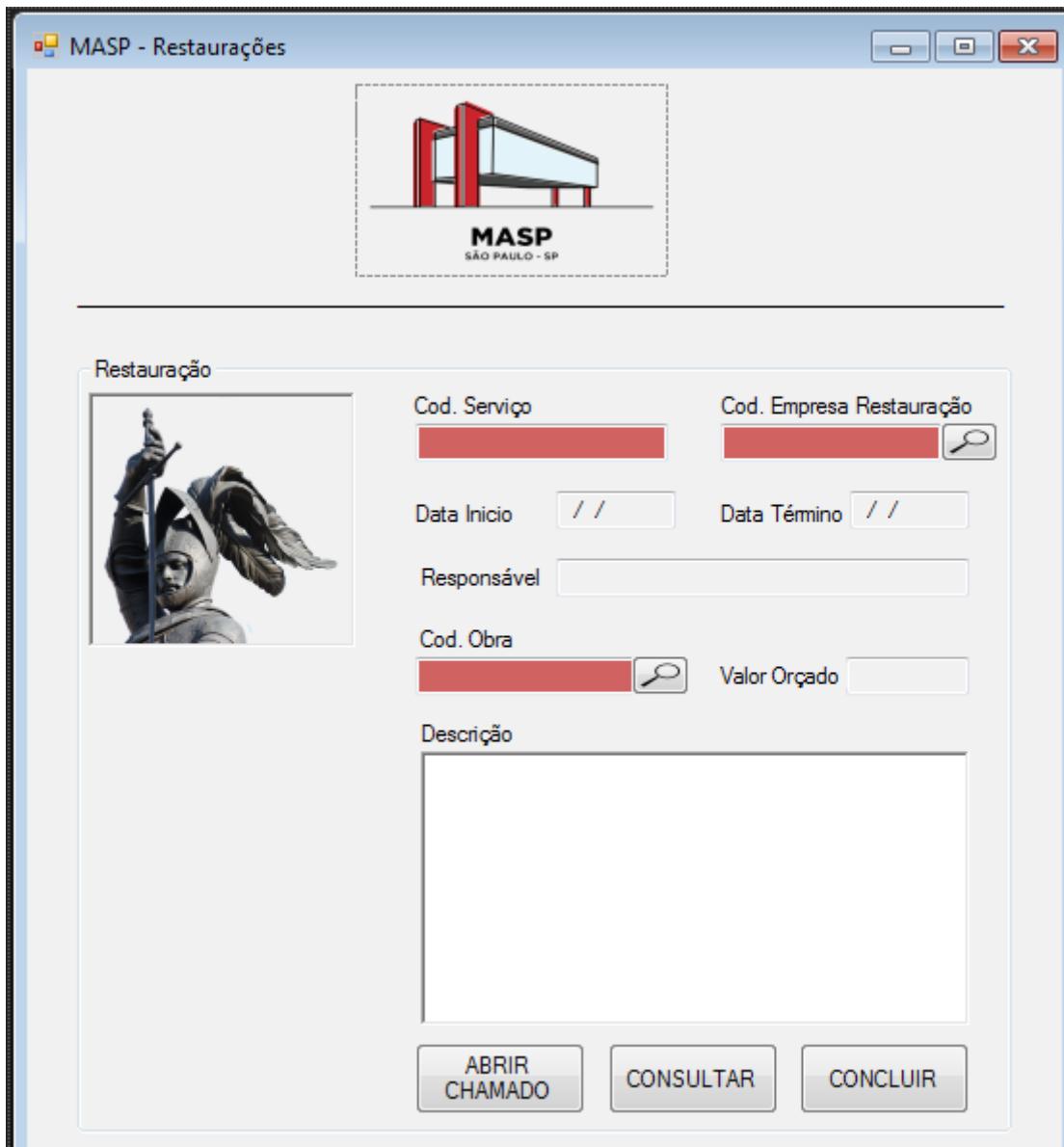


10- Modele uma VCP para o CSU03, utilizando a categorização BCE. A classe de controle deve apresentar dois métodos no mínimo e as classes de entidade devem apresentar seus devidos atributos e métodos. Faça também o protótipo de interface de usuário para a classe <>boundary>< do CSU03.

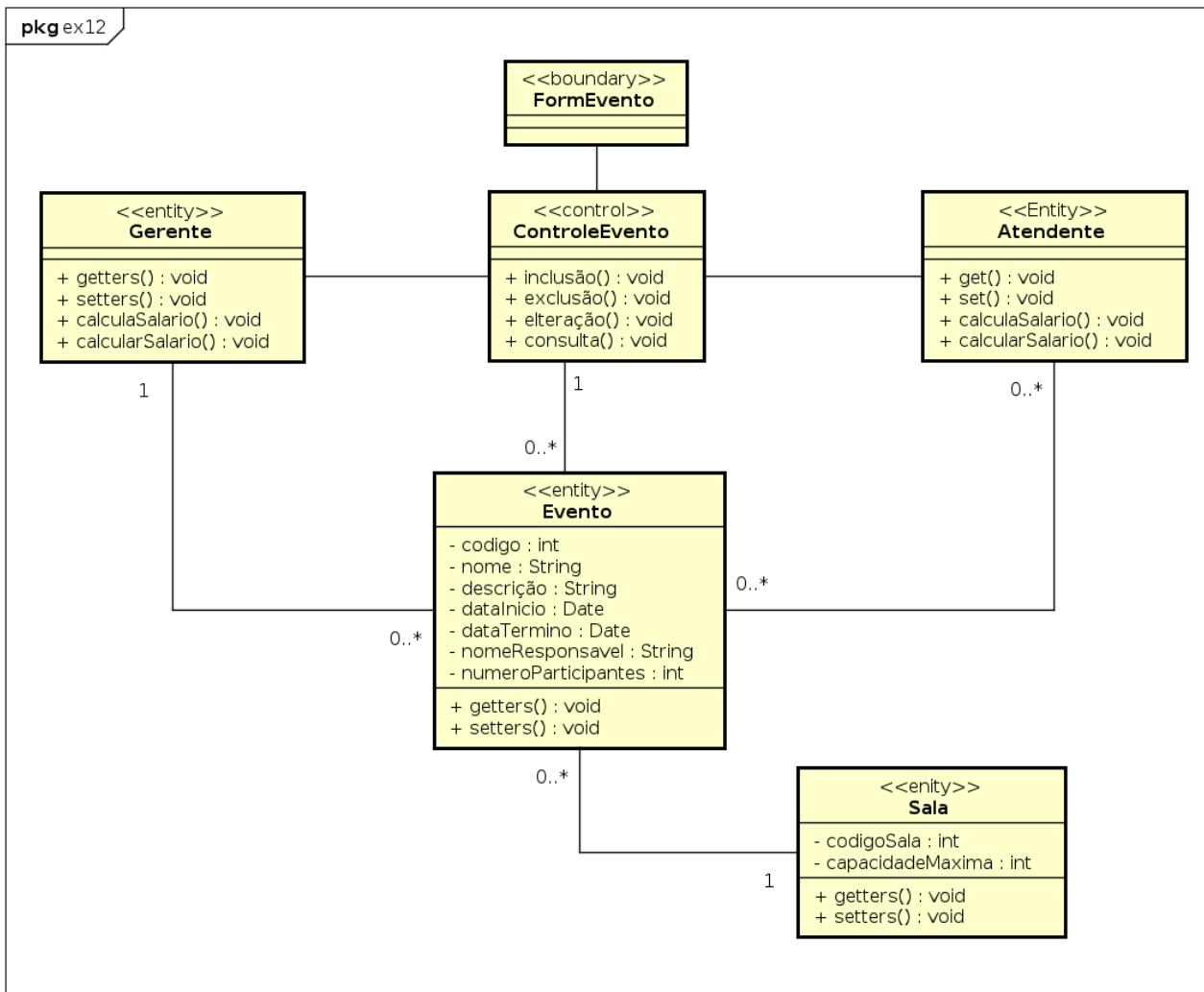


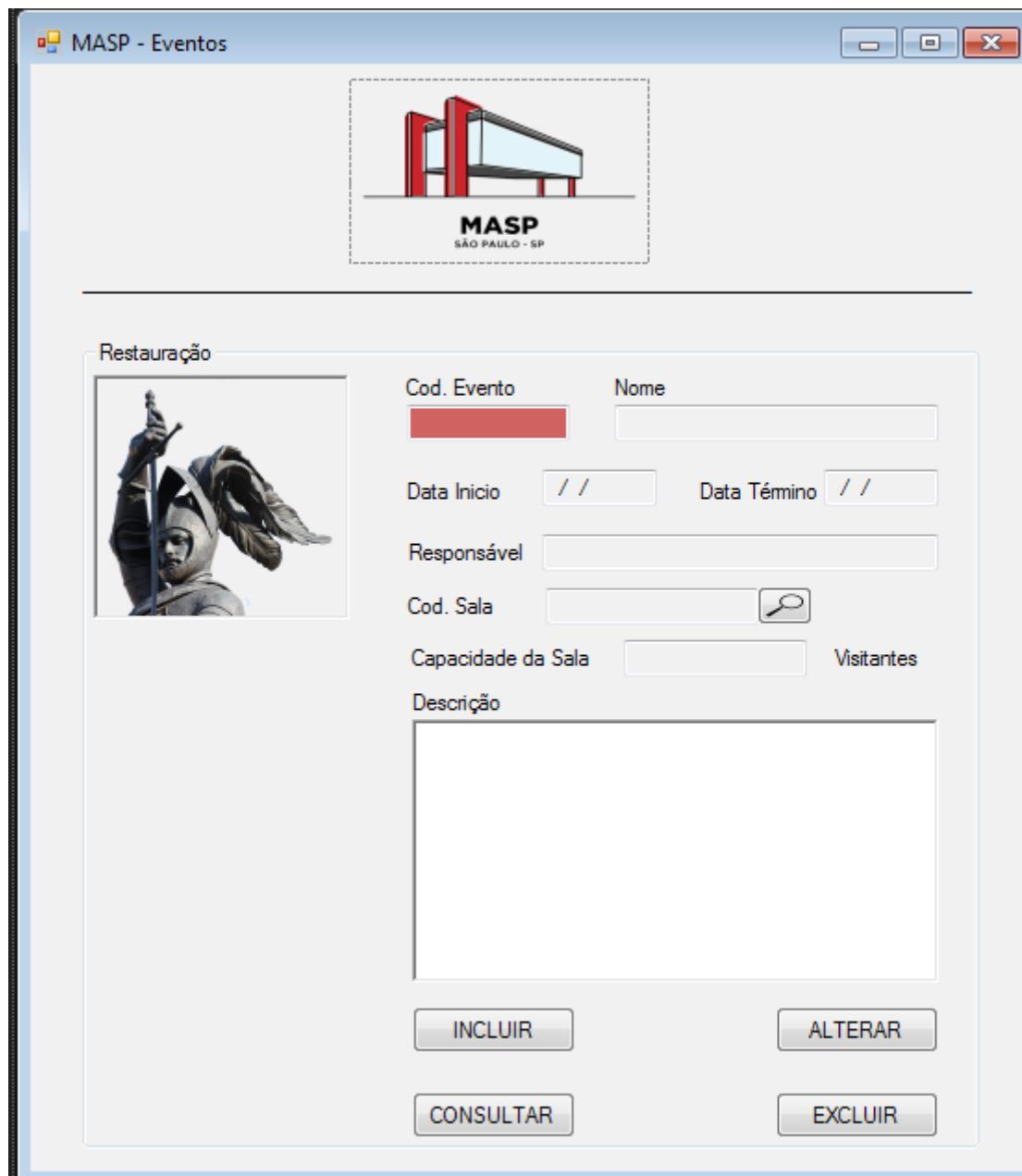
11- Modele uma VCP para o CSU04, utilizando a categorização BCE. A classe de controle deve apresentar dois métodos no mínimo e as classes de entidade devem apresentar seus devidos atributos e métodos. Faça também o protótipo de interface de usuário para a classe <>boundary>> do CSU04.



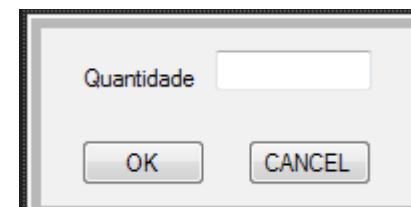
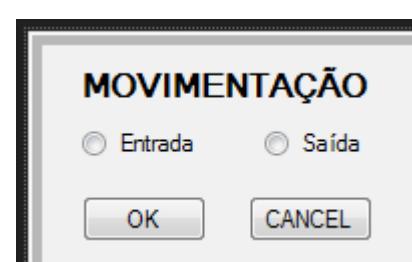
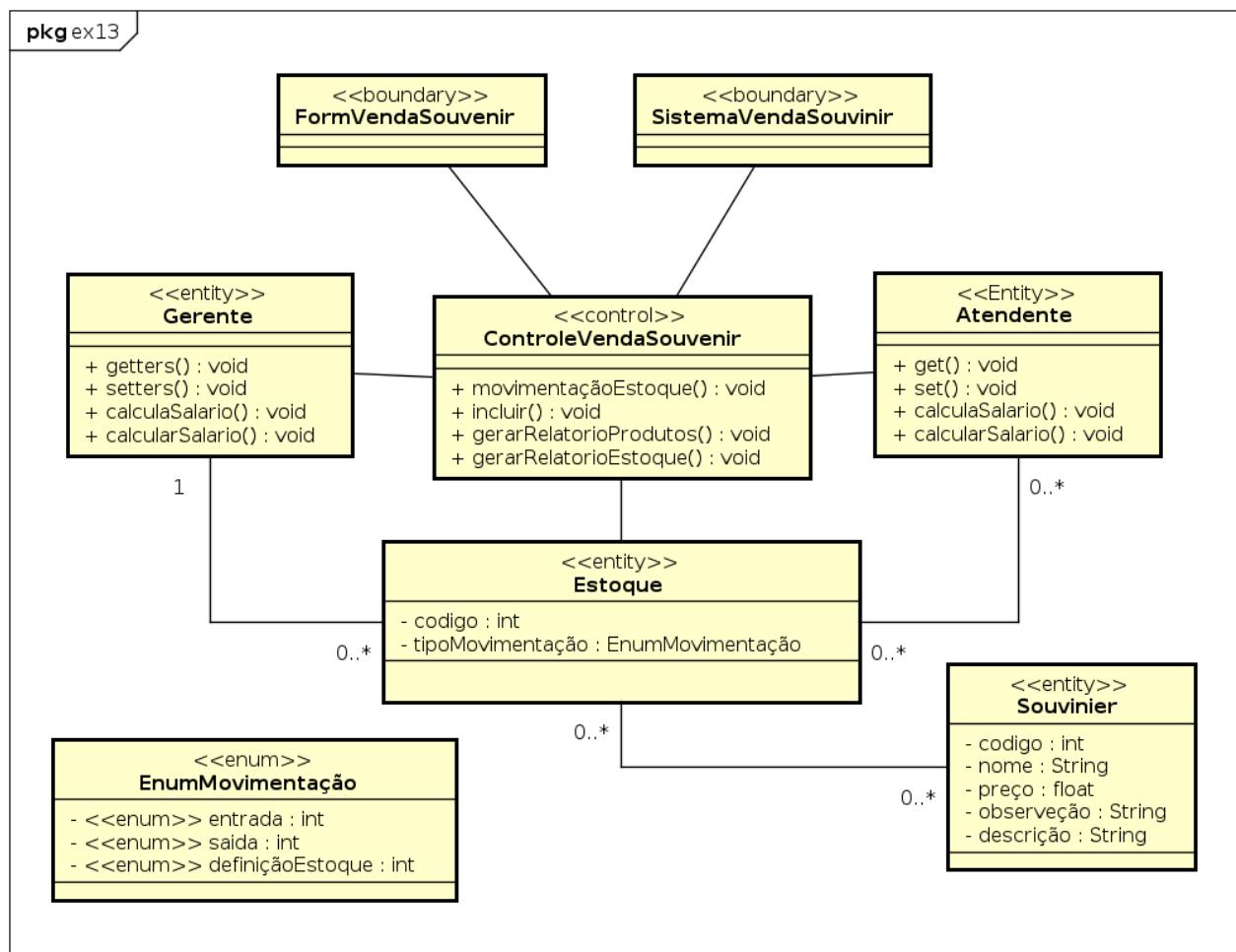


12- Modele uma VCP para o CSU05, utilizando a categorização BCE. A classe de controle deve apresentar dois métodos no mínimo e as classes de entidade devem apresentar seus devidos atributos e métodos. Faça também o protótipo de interface de usuário para a classe <<boundary>> do CSU05.

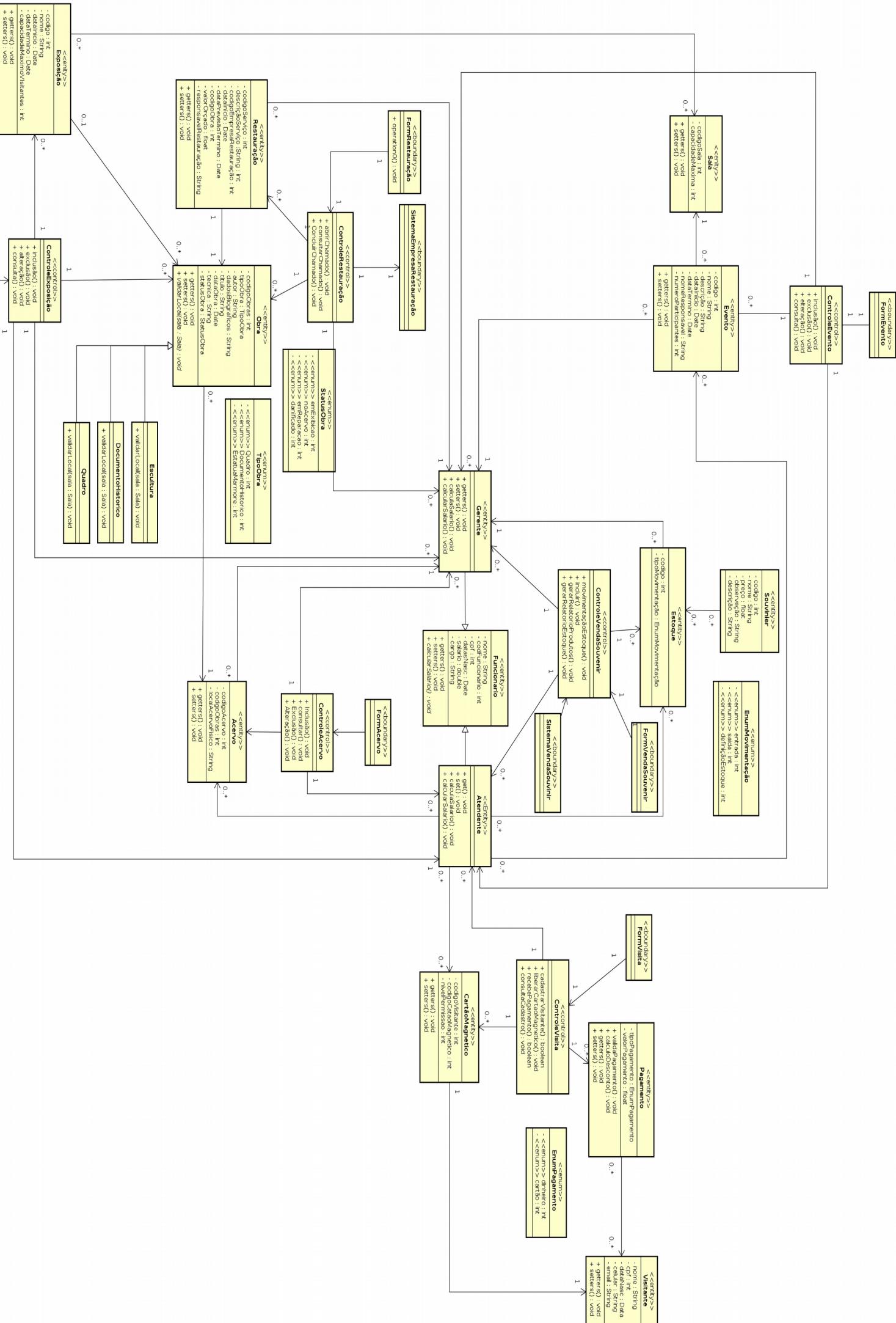




13- Modele uma VCP para o CSU06, utilizando a categorização BCE. A classe de controle deve apresentar dois métodos no mínimo e as classes de entidade devem apresentar seus devidos atributos e métodos. Faça também o protótipo de interface de usuário para a classe <>boundary></> do CSU06.



14- 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. As multiplicidades dos relacionamentos devem ser exibidas.



15- Qual é a classe de entidade mais coesa e a menos coesa do diagrama de classes de projeto? Justifique a tua resposta.

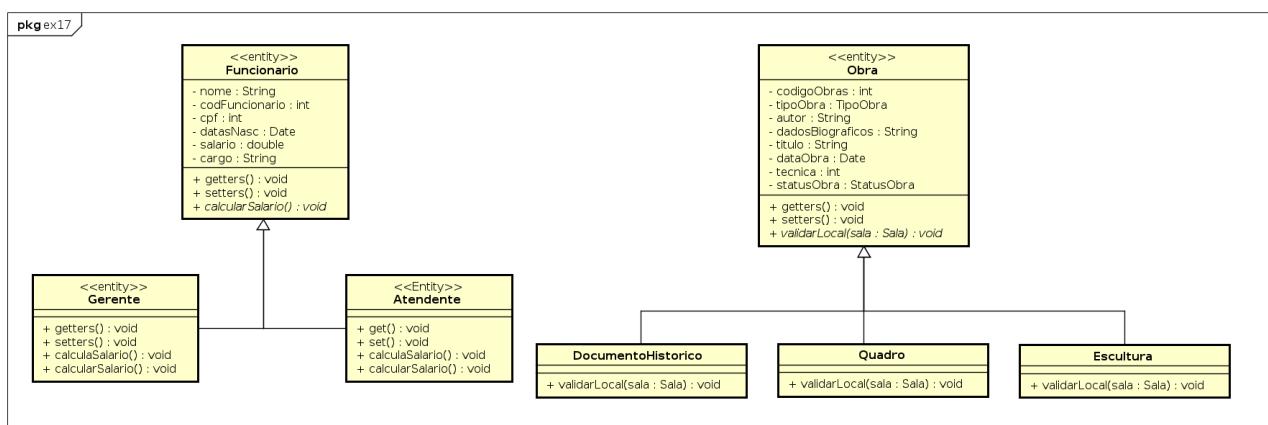
O nível de coesão de todas as classes estão adequadas pois as responsabilidades estão balanceadas entre elas, portanto não há uma com maior ou menor nível de coesão.

16- Qual é a classe de entidade mais acoplada e a menos acoplada do diagrama de classes de projeto? Justifique a tua resposta.

A classe mais acoplada é Gerente, pois as classes ControleRestauracao, Restauracao, ControleExposicao, Acervo, ControleAcervo, ControleVendaSouvenir, Controle Evento, Estoque e Evento dependem dela, pois há necessidade de que o tipo Gerente esteja dentro das classes citadas como registro de a quem pertence ou por quem será utilizada.

Já a classe menos acoplada é Exposição, pois somente o ControleExposição faz CRUD na tipo Exposição.

17- 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. As relações de gen/espec violam o Princípio de Liskov? Justifique a tua resposta.



O Princípio de Liskov não é violado porque Atendente e Gerente são Funcionário. Quadro, Estátua e DocumentoHistórico são todos Obras.

O método calcularSalario() é diferente entre Gerente e Atendente porque o Gerente recebe bonificação pela quantidade de visitas atraídas para o museu enquanto o de Atendente recebe somente o valor do salário.

O método validarLocal(sala: Sala) verifica se é possível colocar determinada Obra na sala em questão por conta do formato da Obra que ocupa dimensões diferentes. Por exemplo um Quadro é quadrado enquanto uma Estátua possui três dimensões.

18- 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 Funcionario {  
  
    private String nome;  
  
    private int codFuncionario;  
  
    private int cpf;  
  
    private Date datasNasc;  
  
    private double salario;  
  
    private String cargo;  
  
  
    public double getSalario()  
    {  
        return salario;  
    }  
  
    public abstract double calcularSalario();  
  
    public String getNome() {  
        return nome;  
    }  
  
    public void setNome(String nome) {  
        this.nome = nome;  
    }  
  
    public int getCodFuncionario() {  
        return codFuncionario;  
    }  
}
```

```

public void setCodFuncionario(int codFuncionario) {
    this.codFuncionario = codFuncionario;
}

public int getCpf() {
    return cpf;
}

public void setCpf(int cpf) {
    this.cpf = cpf;
}

public Date getDataNasc() {
    return datasNasc;
}

public void setDataNasc(Date datasNasc) {
    this.datasNasc = datasNasc;
}

public void setSalario(double salario) {
    this.salario = salario;
}

public String getCargo() {
    return cargo;
}

public void setCargo(String cargo) {
    this.cargo = cargo;
}

}

public class Gerente extends Funcionario {

    @Override
    public double calcularSalario() {
        double comissao=(0.1*Visitante.getTotalVisitantesDia()); //Salario baseado
        em comissão
        return getSalario()+comissao;
    }

}

```

```
public class Atendente extends Funcionario{  
    @Override  
    public double calcularSalario() {  
        return getSalario();  
    }  
}
```

```
public abstract class Obra {  
  
    private int codigoObras;  
  
    private String tipoObra;  
  
    private String autor;  
  
    private String dadosBiograficos;  
  
    private String titulo;  
  
    private Date dataObra;  
  
    private int tecnica;  
  
    private String localAcervoFisico;  
  
    private String statusObra;
```

```
public abstract void validarLocal(Sala sala);  
  
public int getCodigoObras() {  
    return codigoObras;  
}  
  
public void setCodigoObras(int codigoObras) {  
    this.codigoObras = codigoObras;  
}  
  
public String getTipoObra() {  
    return tipoObra;  
}
```

```
public void setTipoObra(String tipoObra) {
    this.tipoObra = tipoObra;
}

public String getAutor() {
    return autor;
}

public void setAutor(String autor) {
    this.autor = autor;
}

public String getDadosBiograficos() {
    return dadosBiograficos;
}

public void setDadosBiograficos(String dadosBiograficos) {
    this.dadosBiograficos = dadosBiograficos;
}

public String getTitulo() {
    return titulo;
}

public void setTitulo(String titulo) {
    this.titulo = titulo;
}

public Date getDataObra() {
    return dataObra;
}

public void setDataObra(Date dataObra) {
    this.dataObra = dataObra;
}

public int getTecnica() {
    return tecnica;
}

public void setTecnica(int tecnica) {
    this.tecnica = tecnica;
}

public String getLocalAcervoFisico() {
    return localAcervoFisico;
}

public void setLocalAcervoFisico(String localAcervoFisico) {
```

```

        this.localAcervoFisico = localAcervoFisico;
    }

    public String getStatusObra() {
        return statusObra;
    }

    public void setStatusObra(String statusObra) {
        this.statusObra = statusObra;
    }

}

public class DocumentoHistorico extends Obra{

    @Override
    public void validarLocal(Sala sala) {
        //Verifica se o há monitores na sala e se possível transfere a imagem ao
mesmo
    }

}

public class Estatua extends Obra {

    @Override
    public void validarLocal(Sala sala) {
        //Verifica se o documento pode ser posto na sala e se pode o coloca
    }

}

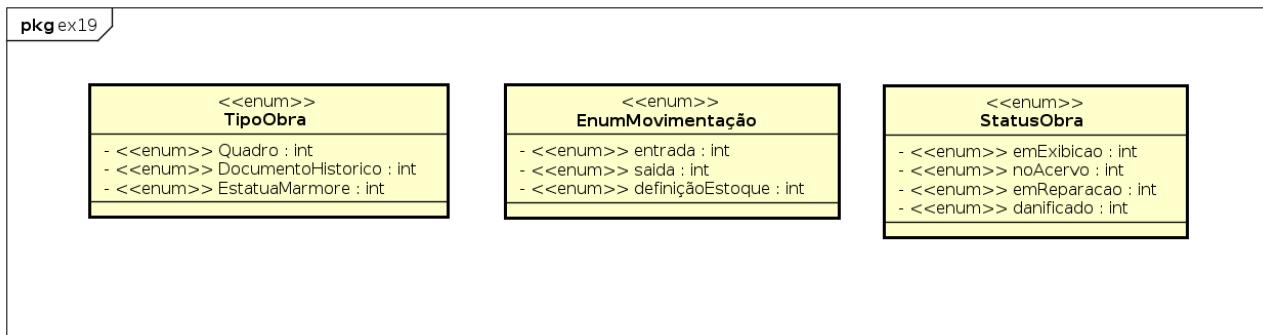
public class Quadro extends Obra{

    @Override
    public void validarLocal(Sala sala) {
        //Verifica se há moldura na sala e registra o quadro na sala se possível
    }

}

```

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



TipoObra é utilizado para dizer qual o tipo obra sem necessidade de criar um classe específica para tal.

EnumMovimentação é necessário para especificar qual tipo de operação, pré definida, pode ser realizada.

StatusObra é necessário para saber o que é possível fazer com Obra, tal como colocar em exibição ou mandá-la para reparação.

20- Apresente a estrutura básica de código em JAVA, C# ou C++ para implementar as três classes enumeradas.

```

public enum TipoObra {
    Quadro, DocumentoHistorico, Estatua
}

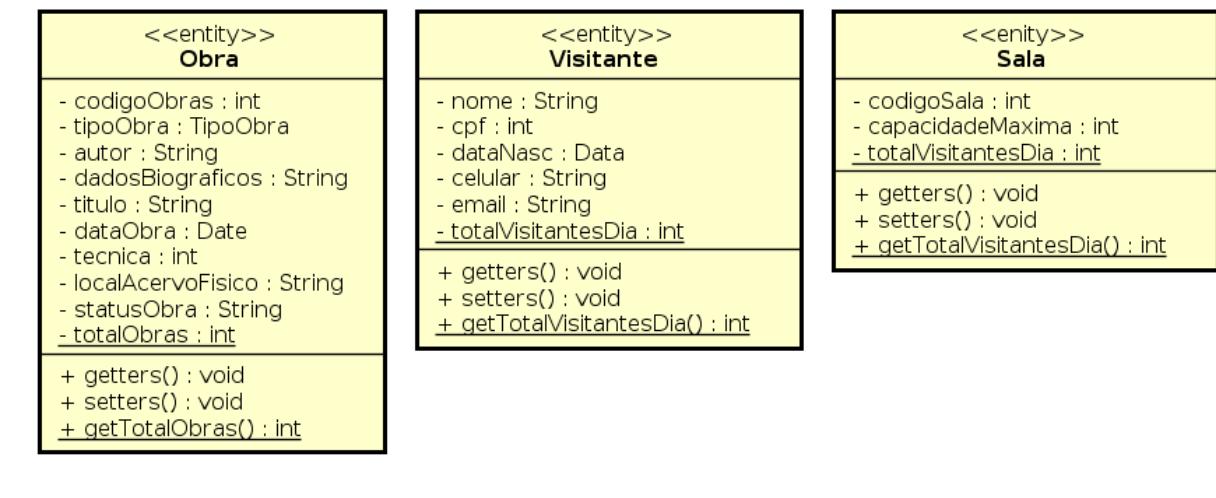
public enum NivelPermissao {
    Almoxarifado, Limpeza, Total, Exposicao
}

public enum StatusObra {
    EmExibicao, NoAcervo, EmReparacao, Danificado
}

```

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

pkg ex21



O totalObras em Obra é utilizado para saber a quantidade total de Obras que tem atualmente que é utilizado em relatórios.

O totalVisitantesDia em Visitante é utilizado para saber o fluxo diário de visitantes no museu.

O totalVisitantesDia em Sala é utilizado para saber quantos visitantes entraram em determinada sala para saber o interesse geral do público.

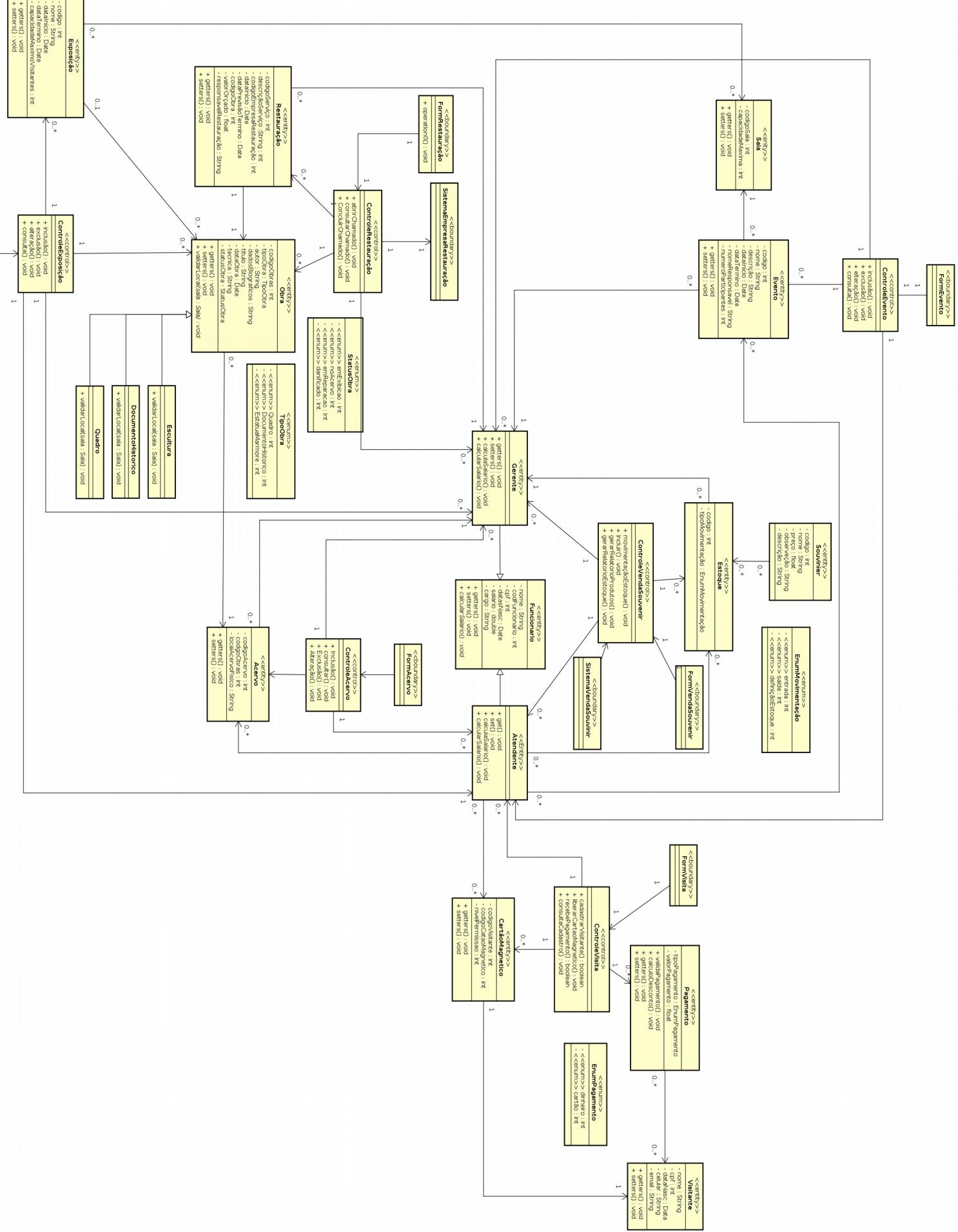
22- Apresente a estrutura básica de código em JAVA, C# ou C++ para implementar os seis membros estáticos.

```
public class Obra {  
  
    private int codigoObras;  
  
    private TipoObra tipoObra;  
  
    private String autor;  
  
    private String dadosBiograficos;  
  
    private String titulo;  
  
    private Date dataObra;  
  
    private int tecnica;
```

```
private String localAcervoFisico;  
private String statusObra;  
private static int totalObras;  
public void getters() {  
}  
public void setters() {  
}  
public static int getTotalObras() {  
    return totalObras;  
}  
}  
  
public class Sala {  
    private int codigoSala;  
    private int capacidadeMaxima;  
    private static int totalVisitantesDia;  
    public void getters() {  
}  
    public void setters() {  
}  
    public static int getTotalVisitantesDia() {  
        return totalVisitantesDia;  
    }  
}  
  
public class Visitante {  
    private String nome;  
    private int cpf;
```

```
private Date dataNasc;  
private String celular;  
private String email;  
private static int totalVisitantesDia;  
public void getters() {  
}  
public void setters() {  
}  
public static int getTotalVisitantesDia() {  
    return totalVisitantesDia;  
}  
}
```

23- Transforme todos os relacionamentos de associação ou agregação entre as classes de entidade e todos os relacionamentos de associação entre as classes de fronteira e controle para dependências estruturais. Explique a vantagem e desvantagem desse tipo de dependência.



A vantagem da dependência estrutural é que ela é mais fácil para implementar, porém por outro lado ela aumenta muito mais o acoplamento do que a dependência não estrutural.

24- Apresente a estrutura básica de código em JAVA, C# ou C++ para implementar as dependências estruturais.

```
//Estrutura básica de algumas classes
```

```
public class Acervo {  
    private int codigoAcervo;  
    private int codgoObras;  
    private String localAcervoFisico;  
    private Gerente gerente;  
    public void getters() {  
    }  
    public void setters() {  
    }  
}  
  
public class Acervo {  
    private int codigoAcervo;  
    private int codgoObras;  
    private String localAcervoFisico;  
    private Gerente gerente;  
    public void getters() {  
    }  
    public void setters() {  
    }
```

```
}

}

public class CartaoMagnetico {

    private int codigoVisitante;

    private int codigoCartaoMagnetico;

    private int nivelPermissao;

    private Visitante visitante;

    public void getters() {

    }

    public void setters() {

    }

}

public class ControleAcervo {

    private Gerente[] gerente;

    private Atendente atendente;

    private Acervo acervo;

    public void incluir() {

    }

    public void consultar() {

    }

    public void exclusao() {

    }

    public void alteracao() {

    }

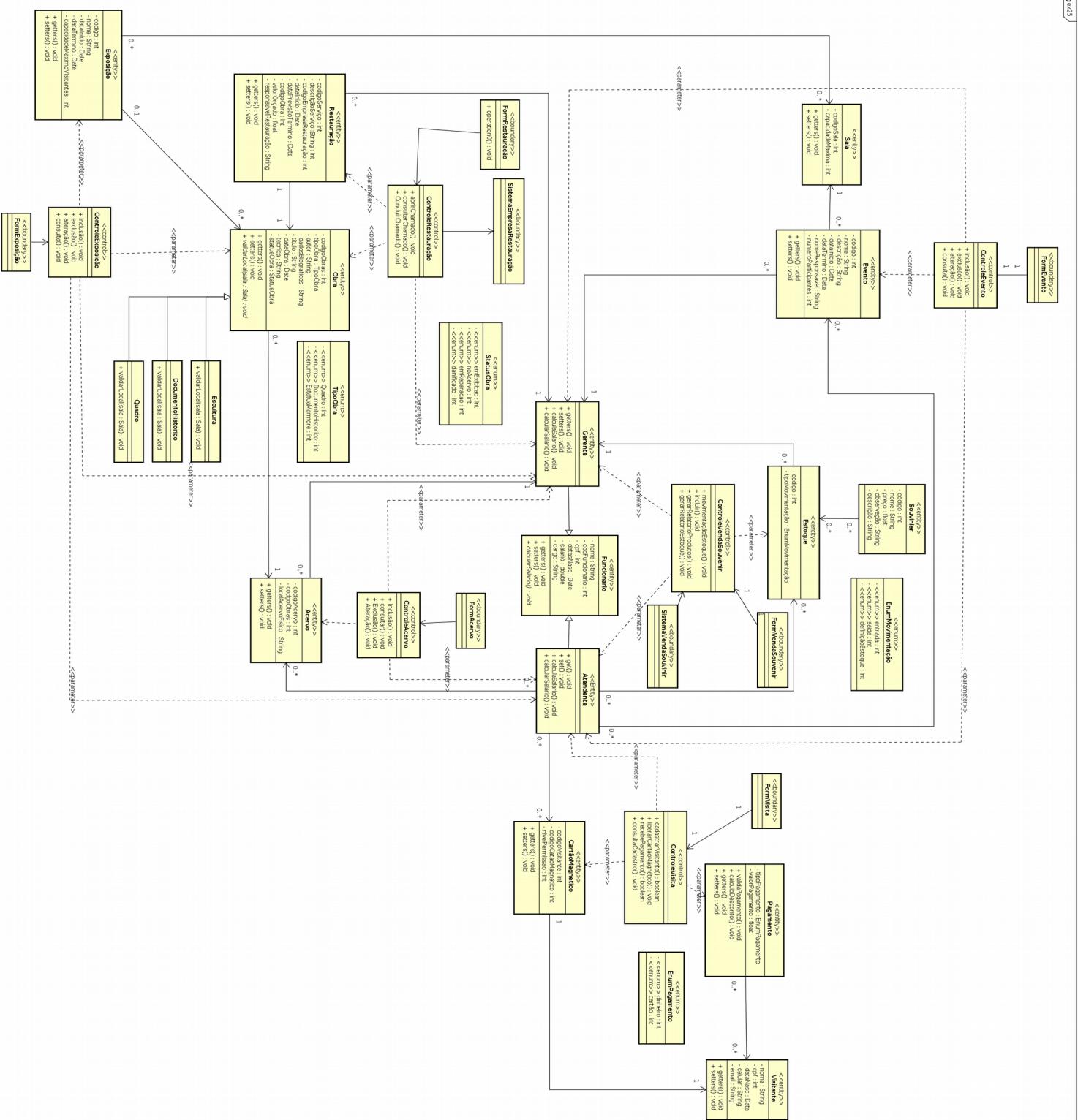
    public void getters() {

    }

}
```

```
public void setters() {  
}  
}
```

25- Transforme todos os relacionamentos de associação entre as classes de controle e entidade para dependências não estruturais por parâmetro. Explique a vantagem e desvantagem desse tipo de dependência.



A vantagem da dependência não estrutural é que faz com que o encapsulamento de cada classe aumente e diminua o acoplamento entre as classes. A desvantagem é que se houver alguma alteração no modelo independente pode ocorrer uma mudança no modelo dependente.

26- Apresente a estrutura básica de código em JAVA, C# ou C++ para implementar as dependências não estruturais por parâmetro.

```
public class ControleVendaSouvenir {  
  
    private Estoque[] estoque;  
  
    private Gerente[] gerente;  
  
    private Atendente[] atendente;  
  
    public void movimentaçãoEstoque(Atendente a, Estoque e) {  
  
    }  
  
    public void incluir(Estoque e) {  
  
    }  
  
    public void gerarRelatorioProdutos(Estoque e, Gerente g) {  
  
    }  
  
    public void gerarRelatorioEstoque(Estoque e, Gerente g) {  
  
    }  
  
}  
  
public class ControleEvento {  
  
    private Gerente[] gerente;  
  
    private Atendente atendente;  
  
    public void inclusão() {  
  
    }  
  
    public void exclusao(Evento evt, Gerente g) {  
  
    }
```

```
}

public void alteração(Evento evt, Atendente a) {

}

public void consulta(Evento evt) {

}

}

public class ControleVisita {

    private CartaoMagnetico[] cartaoMagnetico;

    private Pagamento[] pagamento;

    private Atendente[] atendente;

    public boolean cadastrarVisitante(Atendente a) {
        return false;
    }

    public void liberarCartaoMagnetico(CartaoMagnetico c) {

    }

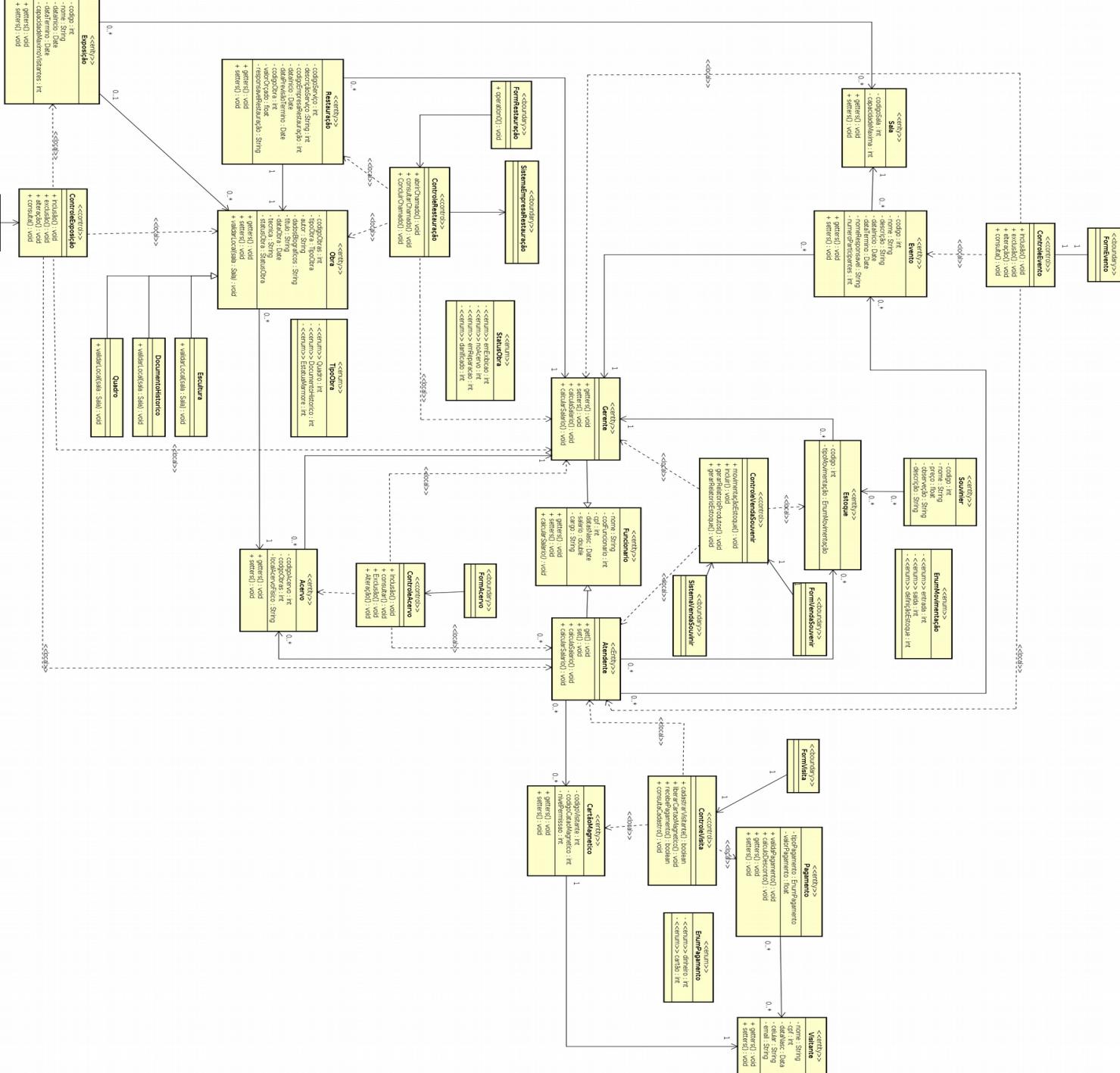
    public boolean recebePagamento(Pagamento p) {
        return false;
    }

    public void consultaCadastro(CartãoMagnetico c) {

    }

}
```

27- Transforme todos os relacionamentos de associação entre as classes de controle e entidade para dependências não estruturais por variável local. Explique a vantagem e desvantagem desse tipo de dependência.



Aumenta o encapsulamento de cada classe e diminuir a acoplamento entre as classes. Quanto menos dependências houver no modelo de classes, maior é o encapsulamento e menor o acoplamento.

28- Apresente a estrutura básica de código em JAVA, C# ou C++ para implementar as dependências não estruturais por variável local.

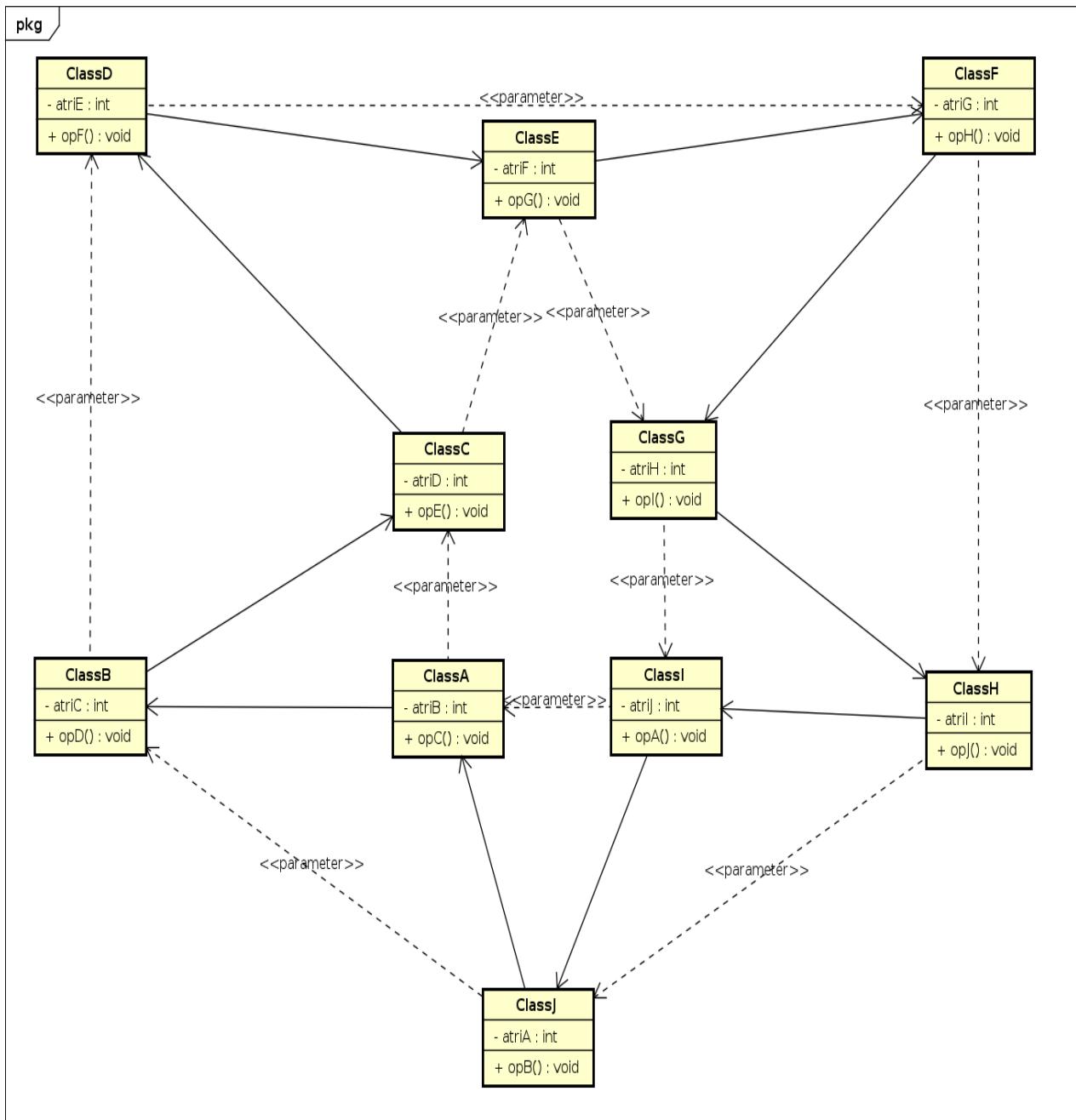
```
public class ControleEvento {  
  
    private Atendente atendente;  
  
    public void inclusao() {  
        Gerente g = Gerente();  
        Evento e = new Evento()  
  
    }  
  
    public void exclusao() {  
        Gerente g = Gerente();  
        Evento e = new Evento()  
    }  
  
    public void alteração() {  
        Evento e = new Evento()  
  
    }  
  
    public void consulta() {  
        Evento e = new Evento()  
    }  
}  
  
public class ControleRestauração {  
  
    public void abrirChamado() {  
        Gerente g = new Gerente;  
        Obra o = new Obra;  
    }  
  
    public void consultarChamado() {  
        Atendente a = new Atendente();  
        Restauracao = new Restauracao();  
    }  
}
```

```

public void ConcluirChamado() {
    Atendente a = new Atendente();
    Restauracao = new Restauracao();
}

```

Parte B



Visão do exercício de abstração:

O exercício é composto por um conjunto de 10 classes, da letra “A“até a “J” onde cada classe que é composta de uma letra possui 4 relações e todas relações, são dependencias estruturais didirecionais

Solução

A solução proposta que não afete a estrutura aumentando as chances de erros foi de transformar as relações que dependem de atributos como unidirecionais e as relações que dependem de métodos como dependencias não estruturais